CUADERNO PARA DOCENTES

Actividades para aprender a Program.AR

Segundo Ciclo de la **Educación Primaria**
y Ciclo Básico de la **Educación Secundaria**

TERCERA EDICIÓN
Febrero de 2025

<Program.AR/> Fundación SADOSKY CAF BANCO DE DESARROLLO DE AMÉRICA LATINA Y EL CARIBE

Actividades para aprender a Program.AR

Segundo Ciclo de la Educación **Primaria** y Ciclo Básico de la Educación **Secundaria**

<Program.AR/> Fundación SADOSKY CAF BANCO DE DESARROLLO DE AMÉRICA LATINA Y EL CARIBE

Autores

Tomás Leonel Caballero
Fernando Gastón Cáceres
Javier Castrillo
Julián Dabbah
Alfredo Héctor Sanzo (Alf)

Coordinación autoral

Julián Dabbah

Coordinación editorial

Inés Roggi

Edición

Florencia N. Acher Lanzillotta

Diseño

Fabio Viale
Lucia Rovira

Ilustración de personajes

Tony Ganem

Fundación Sadosky

Directora Iniciativa Program.ar
Mara Borchardt

Director Ejecutivo Fundación Sadosky
Fernando Schapachnik

Actividades para aprender a Program.AR : segundo ciclo de la educación primaria y ciclo básico de la educación secundaria / Tomás Leonel Caballero ... [et al.] ; Coordinación general de Inés Roggi ; Juli Dabbah ; Editado por Florencia Acher ; Ilustrado por Tony Ganem. - 3a ed mejorada. - Ciudad Autónoma de Buenos Aires : Fundación Sadosky, 2025.
Libro digital, PDF

Archivo Digital: descarga y online
ISBN 978-987-48926-3-8

1. Educación Tecnológica. I. Caballero, Tomás Leonel II. Roggi, Inés, coord. III. Dabbah, Juli, coord. IV. Acher, Florencia , ed. V. Tony Ganem, ilus.
CDD 607

Cómo citar este documento:

Fundación Sadosky (2025), *Cuaderno para docentes. Actividades para aprender a Program.AR. Segundo Ciclo de la Educación Primaria y Ciclo Básico de la Educación Secundaria*. Buenos Aires. Disponible en: https://program.ar/

Prólogo

¿Por qué enseñar programación en la escuela argentina? En un mundo cada vez más atravesado por la tecnología, esta pregunta se convierte en una reflexión fundamental. La programación no solo permite interactuar con las herramientas digitales que nos rodean, sino también entenderlas, modificarlas y crearlas. No se trata únicamente de enseñar a usar dispositivos, sino de abrir la "caja negra" de la computación, permitiendo a las y los estudiantes comprender los principios que la hacen funcionar y, a partir de ellos, ser capaces de crear nueva tecnología. Saber programar no es solo una habilidad técnica, sino una herramienta para pensar, analizar y resolver problemas de manera creativa.

Desde la Iniciativa Program.AR sostenemos que comprender las ciencias de la computación es esencial para descifrar las lógicas tecnológicas que median nuestras interacciones con el mundo. La escuela, como espacio de formación de ciudadanía, tiene la responsabilidad de preparar a las nuevas generaciones para enfrentar los desafíos de un entorno cada vez más digitalizado. Por ese motivo, impulsamos desde 2013 la inclusión de las ciencias de la computación en los niveles obligatorios de la educación en la Argentina, entendiendo que para la construcción de estos saberes no basta con estar en contacto con dispositivos computacionales. Al igual que sucede con otras herramientas, su mero uso no necesariamente revela sus lógicas internas de funcionamiento y menos aún los principios básicos en los que estas se fundan y, por lo tanto, es necesario que se conviertan en objeto de enseñanza.

El debate global sobre la enseñanza de la programación refleja un reconocimiento creciente de su importancia. Sin embargo, también evidencia que no hay un modelo único o universalmente exitoso. Esta tarea no puede abordarse replicando experiencias de otros países sin considerar las particularidades de cada país. Cada contexto educativo, social y cultural requiere estrategias diseñadas para responder a sus necesidades específicas. Por eso, Program.AR trabaja desde una perspectiva federal, respetando las autonomías provinciales y promoviendo la inclusión de rasgos culturales de la Argentina en la enseñanza de la programación.

Uno de los pilares fundamentales para lograr esta transformación es la formación docente. Enseñar programación requiere de educadores preparados, no solo en términos técnicos, sino también con herramientas pedagógicas adecuadas para compartir estos conocimientos de manera efectiva y significativa. En Program.AR, estamos comprometidos con generar capacidades locales a través de la capacitación docente, tanto inicial como continua. La reedición de este cuaderno renueva el compromiso con ese esfuerzo, ofreciendo recursos prácticos que buscan inspirar y guiar a las y los docentes en la implementación de secuencias didácticas innovadoras en el aula.

Entendemos que enseñar programación desde los primeros niveles educativos no significa formar programadores. El objetivo de esta iniciativa es mucho más amplio: queremos formar una ciudadanía crítica y consciente de cómo la tecnología impacta en la vida. Queremos que

las y los estudiantes de la Argentina superen el rol del simple consumo tomando decisiones informadas, entendiendo las implicancias sociales de las herramientas digitales e, incluso, creando activamente tecnología.

Atentos a las desigualdades en la apropiación de estos saberes y con la convicción y la certeza de que la escuela debe y puede mitigarlas y revertirlas, impulsamos su incorporación desde edades tempranas para desarmar preconceptos y estereotipos que alejan de las ciencias en general (y de la computación en particular) a grupos sociales como el de las mujeres y el colectivo LGBTIQ+. Pensamos este cuadernillo como un recurso pedagógico hacia una enseñanza inclusiva y transformadora para formar, sin distinciones, ciudadanas y ciudadanos. Es, en definitiva, una invitación a las y los docentes de todo el país a sumarse a un desafío urgente y necesario de transformación educativa.

El equipo de Program.AR

Índice

Introducción

La programación es el área dentro de las Ciencias de la Computación (CC) que estudia los modos en que las personas podemos definir el comportamiento deseado de las computadoras. Por ejemplo, los videojuegos, las aplicaciones de los dispositivos móviles o los sistemas digitales con los que interactuamos como parte de nuestra vida cotidiana (como los que administran el registro de nuestros datos personales o la historia académica de nuestros estudiantes) son ejemplos de programas. La complejidad de los programas y sus características varían, pero todos tienen en común ideas, herramientas y principios fundamentales que podemos aprender sin ser especialistas en programación.

Este cuaderno tiene como objetivo ser una guía para que las y los docentes puedan enseñar programación a estudiantes del Segundo Ciclo de la Educación Primaria y el Ciclo Básico de la Educación Secundaria. Para facilitar la planificación y el trabajo en clase, los contenidos se presentan como secuencias didácticas formadas por actividades. Cada unidad incluye secuencias en las que se presentan, se profundizan y se refuerzan conceptos de programación.

El ejercicio de la programación requiere no solo saberes teóricos, sino también prácticas que involucran planificación, generalización, interpretación y puesta a prueba. Esto hace que su aprendizaje demande una práctica intensa y sostenida en el tiempo, y que se organice en un recorrido espiralado en el que los contenidos se aborden en sucesivas capas de complejidad y se integren con lo ya aprendido. Tanto la organización de las actividades dentro de las secuencias como el orden en el que estas aparecen a lo largo del cuaderno están pensados con este enfoque progresivo: cada unidad comienza con desafíos simples cuyo objetivo es motivar el contenido en cuestión y propone un recorrido de aprendizaje que concluye con su aplicación para hacer algo nuevo de mayor complejidad; cada contenido nuevo se presenta en relación con las ideas fundamentales abordadas en las secuencias anteriores, explicitando cómo se integra con ellas y cómo este amplía las posibilidades que brindaban las otras[1].

Para llevar a cabo esta tarea, se utilizan dos aplicaciones diseñadas especialmente para la enseñanza de la programación: Pilas Bloques y Scratch. **Pilas Bloques** es una herramienta diseñada para enseñar a programar mediante desafíos con diversos niveles de dificultad para introducir a las y los estudiantes al mundo de la programación por medio de bloques. Esta aplicación, desarrollada en la Argentina por la Iniciativa Program.AR de la Fundación Sadosky, puede descargarse e instalarse o utilizarse online gratuitamente en https://pilasbloques.program.ar/. Por otro lado, **Scratch** es un entorno de enseñanza de la programación que permite crear historias interactivas, animaciones, juegos y música, compartiéndolas en la web, y formar parte de una numerosa comunidad de estudiantes y docentes distribuida por todo el planeta. Está disponible para descargarse e instalarse o utilizarse online gratuitamente en https://scratch.mit.edu.

[1] En la portada de cada secuencia, se puede consultar cuáles son los conceptos nuevos y los saberes previos requeridos.

Una estrategia didáctica para enseñar programación: la enseñanza por indagación

El modelo de enseñanza por indagación surge, de algún modo, como una reacción frente al modelo de enseñanza tradicional. Ya a principios del siglo XX, y refiriéndose a la enseñanza de las Ciencias Naturales, John Dewey advertía que había un énfasis en la acumulación de la información en las ciencias naturales y una desatención de una manera de pensar y una actitud de la mente propias de las ciencias[2].

Podemos situar a la enseñanza por indagación en el marco de las estrategias didácticas que fomentan el aprendizaje por descubrimiento, el aprendizaje basado en problemas y el aprendizaje basado en proyectos. La enseñanza por indagación no es un modelo didáctico nuevo, está presente en documentos curriculares hace más de dos décadas. En 2012, Mariela Collo y otros autores observaban: "En los documentos curriculares y en el ámbito educativo en general, existe un consenso acerca de la utilidad de esta metodología de enseñanza. En nuestro país, los Núcleos de Aprendizajes Prioritarios[3] prescriben diferentes situaciones de enseñanza enmarcadas en la indagación escolar. Este enfoque recibe diferentes nombres, como modelo de investigación escolar, enseñanza por investigación o investigaciones orientadas"[4].

Antecedentes

A mediados del siglo XX, en el marco de la carrera espacial que llevaron adelante la Unión Soviética y Estados Unidos durante la Guerra Fría y producto de la percepción de la falta de preparación científica de la población general, Estados Unidos comienza una fuerte revisión de la calidad de la enseñanza de las ciencias en la escuela y lleva adelante una reforma curricular que se extiende en el mundo anglosajón.[5] En este contexto, Joseph Schwab alentaba a los docentes de ciencias a "enseñar en un formato de indagación"[6]. La indagación científica no solo se aborda como contenido y como habilidad a desarrollar en las y los estudiantes, sino también como estrategia didáctica que provee metodologías y estructuras que son consistentes con la forma en que las personas hacen y aprenden ciencia. Para que los y las estudiantes construyan los conceptos a partir de su propia experiencia de este modo, la enseñanza busca enfrentarlos a situaciones problemáticas y se lleva adelante a través del uso contextualizado de la terminología apropiada, usando ejemplos, analogías, modos de ilustrar los conceptos en

[2] Furman, M. y de Podestá, M. E. (2010). *La aventura de enseñar Ciencias Naturales*. AIQUE Educación, 52.
[3] Consejo Federal de Cultura y Educación (2004). Núcleos de Aprendizaje Prioritarios: Ministerio de Educación, Ciencia y Tecnología
[4] Collo, M., De la Fuente, C., Gabaroni, B., Gianatiempo, A., Israel, G., Melo, S., Podestá, M. E., Rosenzvit, M., y Seará, V. (2012). "Ciencias Naturales. Material para Directivos de Educación Primaria", en M. Furman, P. Salomón, y A. Sargorodschi (eds.), *Programa para el acompaniamiento y la mejora escolar.* IIPE - UNESCO, 8. http://servicios2.abc.gov.ar/lainstitucion/organismos/programa_para_el_acompaniamiento_y_la_mejora_escolar/materiales_de_trabajo/directores/ciencias_naturales.pdf
[5] Adúriz-Bravo, A., y Izquierdo Aymerich, M. (2002). "Acerca de la didáctica de las ciencias como disciplina autónoma", en *REEC: Revista electrónica de enseñanza de las ciencias*, 1(3), 130-140.
[6] Schwab, J. (1960). "Enquiry, the science teacher, and the educator", en *The Science Teacher*, 27, 6–11.

un lenguaje comprensible para ellas y ellos. Una vez que comprenden la noción, se le pone el nombre formal al concepto.[7]

Como señalan especialistas en la didáctica de las Ciencias Naturales, "si bien existe un acuerdo sobre la importancia de que los docentes de ciencias utilicen una metodología de enseñanza por indagación, el mayor problema pasa por ponerla en práctica. [...] Los alumnos no aprenden 'Ciencias' (entendidas como producto y como proceso) simplemente aprendiendo términos como 'hipótesis' y 'predicciones' o memorizando los pasos del método científico. Ni tampoco realizando experiencias sin comprender qué están haciendo ni por qué"[8].

En su investigación basada en textos escolares, Clark A. Chinn y Betina A. Malhotra dan cuenta de habituales distorsiones de la indagación escolar: "la indagación que se hace en la escuela hace que los estudiantes vean el razonamiento como algo simple y seguro, y que se haga hincapié en la observación superficial, lo que provoca que las y los estudiantes tengan una creencia distorsionada sobre la ciencia"[9].

El desafío que tenemos como docentes no es sencillo. No basta con brindar buenas explicaciones o argumentos, es necesario lograr que las y los estudiantes se impliquen con el contenido, procesen, consoliden, relacionen, reflexionen. La enseñanza por indagación requiere tiempo y, por ende, demanda una priorización curricular. Diversas investigaciones relevadas por Lloyd H. Barrow señalan que para una o un docente disponer de experiencias de indagación como estudiante resulta más valioso que teorizar sobre ella[10].

La didáctica específica de las CC se nutre de estos desarrollos didácticos de otras ciencias, como las naturales.

La indagación en la enseñanza de la programación

Para que la enseñanza de la programación sea eficaz debe ser abordada desde metodologías que permitan dar cuenta de la complejidad de la tarea (como la descomposición de problemas o la generalización de patrones comunes), y no desde la concepción operacional de "secuencia de pasos" o una receta a seguir. Antes de comenzar a codificar, se hace indispensable la necesidad de reflexionar sobre el problema que se va a resolver.

Esta metodología de enseñanza cobra sentido cuando se busca que las y los estudiantes aprendan y comprendan profundamente conceptos claves y estructurantes que permiten imaginar la automatización de muchas tareas a partir de un mismo concepto. Por ejemplo, la noción de condicional posibilita programar a un Pac-Man para que titile cuando come una cereza; y ese mismo concepto también permite programar una alarma de incendio para que se dispare cuando detecta humo. Para que estas "ideas poderosas" no sean trivializadas, es necesario que cada estudiante "descubra" el potencial que poseen para resolver un problema significativo para

[7] Barrow, L. H. (2006). "A brief history of inquiry: From dewey to standards", en *Journal of Science Teacher Education*, 17(3), 265-278.
[8] Collo, M. *et al.*, *ibid.*
[9] Barrow, L. H., *ibid.*
[10] *Ibid.*

él o ella. Es por ello que la enseñanza por indagación resulta una estrategia útil para favorecer la emergencia de tales ideas e invita a nuestros estudiantes a explorarlas y entenderlas.[11]

La enseñanza por indagación implícita en una propuesta por desafíos "consiste en invitar a los estudiantes a resolver un problema, pero intencionalmente no les ofrecemos todos los conceptos ni los pasos para resolverlo"[12]. Buscamos que los conceptos surjan del trabajo de resolver problemas a través de la exploración y el descubrimiento, y no al revés, ya que, si se exponen los conceptos previamente, estos no tienen razón de ser. Se pretende que las y los estudiantes encuentren el sentido de lo que están aprendiendo antes de saber exactamente el nombre y el concepto involucrados. No buscamos que sean unos hábiles seguidores de pasos dados, sino que construyan los propios.

Además de la exploración y el análisis de problemas, construir estos pasos hacia la solución requiere otra habilidad esencial: la transferencia, es decir, apelar a saberes aprendidos en otros contextos para resolver nuevos problemas. Un ejemplo es reconocer que el Pac-man y la alarma de incendios responden al mismo principio y que, por lo tanto, puede utilizarse la misma herramienta de programación para construir ambos. Si bien este es el fin último de esta enseñanza, muchas veces se pasa por alto este proceso, porque se da por supuesto o se lo deja bajo responsabilidad individual de las y los estudiantes. Sin embargo, es una habilidad compleja que requiere de toda nuestra dedicación como docentes. Pensar en soluciones a problemas similares para establecer analogías con soluciones conocidas es un ejemplo de cómo podemos trabajar explícitamente sobre ella.

Metodología

Una propuesta de enseñanza llevada adelante desde el enfoque por indagación suele comenzar con un proceso de exploración, usualmente poniendo énfasis en el trabajo cooperativo, y termina con la presentación de los conceptos por parte de las y los docentes a través de una puesta en común y la reflexión sobre las actividades realizadas para construir la solución. De esta manera, los conceptos presentados no se anticipan a la práctica, sino que, por el contrario, son motivados mediante una situación problemática que precisa del nuevo concepto para ser resuelta. Su presentación ocurrirá luego de que las y los estudiantes hayan intentado encontrar una solución con los elementos de los que disponían hasta ese momento y hayan identificado qué es lo que todavía no estaban en condiciones de resolver. En este esquema, **las intervenciones docentes tienen más que ver con alentar y orientar la exploración y las reflexiones comunes que con enunciar contenidos**. Por el contrario, estos aparecen como conclusiones de las experiencias y reflexiones de las y los estudiantes.

Esta metodología permite a los y las estudiantes construir un concepto con mayor solidez, pues intentaron solucionar el problema primero y, por lo tanto, les resulta evidente y más significati-

[11] Areces, C., Benotti, L., Cortez, J. J., Fervari, R., García, E., Gómez, M., Martínez, M. C., Onetti, C. M., Rodríguez, E. S., y Wolovick, N. (2018). *Ciencias de la Computación para el aula - Manual para docentes. 2° segundo ciclo primaria.* Fundación Sadosky. http://bit.ly/2uWUDIj

[12] *Ibid.*

vo cómo este participa de la solución. En el desarrollo de una actividad por indagación, solemos reconocer los siguientes momentos:

1. El o la docente propone un **problema** que presenta un desafío a los y las estudiantes.
2. Los y las estudiantes intentan resolver el problema. Esta experiencia les permite **indagar** en la naturaleza del problema apelando a lo que ya conocen e incluso experimentar con nuevas soluciones o herramientas que se les ocurran. El o la docente debe acompañar el proceso observando el progreso y las dificultades de sus estudiantes e intervenir en momentos de frustración para proveer orientación y alentar a que continúen la exploración.
3. El o la docente modera una puesta en común donde se concluye cuál es el problema y **qué haría falta** para poder resolverlo.
4. De acuerdo con esta necesidad, la o el docente presenta la **nueva herramienta o concepto**.
5. Los y las estudiantes vuelven a enfrentarse al problema.
6. Para concluir la actividad, se realiza una nueva **puesta en común**, en la que las y los estudiantes comparten qué hicieron (y por qué) para resolver el problema. La o el docente modera el intercambio para realizar **un resumen o una reflexión de cierre, de la que se desprenda la conceptualización buscada**. Enuncia y pone nombre a los conceptos o herramientas trabajadas y explicita el resultado del proceso de aprendizaje. Si corresponde, también realiza **generalizaciones** y alienta la **transferencia** a otros problemas o desafíos que considere similares.

Enseñar computación desde la mirada de la Educación Sexual Integral (ESI)

La Fundación Sadosky busca propiciar una experiencia educativa inclusiva y promotora de la equidad de género y diversidades. Existe una fuerte desigualdad de género en el acceso al uso de recursos tecnológicos y al conocimiento en el ámbito de las CC. Uno de los motivos de esta brecha tiene que ver con que las ciencias, en general, y las CC en particular fueron consideradas histórica y socialmente disciplinas de varones[13]. Como docentes y miembros de la comunidad de las CC, sostenemos que tenemos una oportunidad en las aulas de contribuir a desnaturalizar este prejuicio y generar estrategias que incentiven la inclusión de estudiantes mujeres y de identidades de género trans y no binarias al campo científico. El género no es el único factor que genera brechas; otros prejuicios asociados a estereotipos sociales también determinan el acceso a este campo del saber.

¿Es posible desarmar estos prejuicios en una clase de programación? ¿Qué cambios podemos hacer en las dinámicas áulicas para que sean más inclusivas? La Educación Sexual Integral de

[13] Ortmann, Cecilia (2019). "Clase virtual 3.G: La enseñanza de la ESI en el aula de Física, Química y Matemática", en *Módulo 3: Enseñanza de la ESI. Especificaciones por nivel y/o por área curricular.* Diplomatura de Extensión en Educación Sexual Integral, Facultad de Filosofía y Letras de la Universidad de Buenos Aires.

la Nación Argentina[14] promueve el derecho de las infancias y las adolescencias a recibir una educación inclusiva, libre y justa. Es una valiosa herramienta de la que disponemos las y los docentes para pensar y proponer experiencias de aprendizaje transformadoras. Esta ley es el marco de algunas estrategias que proponemos en las secuencias didácticas que conforman este cuadernillo. Son propuestas concretas que buscan ser un puntapié para (re)pensar y (re) organizar las clases de computación con una mirada inclusiva.

Empezar por lo que nos pasa como docentes

Cuando enseñamos, ponemos en juego lo que pensamos, sentimos y creemos. Las desigualdades históricas presentes en las ciencias exactas, en general, y en las CC, en particular, generan representaciones y estereotipos de género que hemos naturalizado e incorporado a lo largo de nuestras trayectorias y que podemos reproducir en el trabajo didáctico-pedagógico cotidiano sin darnos cuenta.

Un camino para desarmar estas ideas es recurrir a una de las llamadas *puertas de entrada*[15] de la ESI que promueve una **reflexión personal** sobre nuestras trayectorias educativas y de formación para identificar aquellas valoraciones y creencias relacionadas con los géneros que aprendimos y atender si estamos poniendo en juego alguno de estos estereotipos en los actos pedagógicos.

El currículo implícito y las expectativas de desempeño

Estamos enseñando al impartir contenidos disciplinares y también en los mensajes que dirigimos a las y los estudiantes al comunicarnos. Por eso es importante atender a las posibles diferencias de comportamiento pedagógico y preguntarnos: ¿qué expectativas de rendimiento tenemos con respecto a los estudiantes varones?, ¿esperamos el mismo desempeño académico de estudiantes mujeres o de identidades de género trans y no binarias? Estas preguntas pueden ayudarnos a visibilizar aquellas expectativas de rendimiento basadas en el género de cada estudiante. Al representarse las CC histórica y socialmente como una disciplina de varones y para varones, muchas veces, el éxito académico de los estudiantes varones es concebido como "natural" y esperable. Por el contrario, estudiantes mujeres o de identidades de género trans y no binarias tienen la obligación de probar que pueden hacerlo; y cuando lo logran, el éxito se explica gracias al esfuerzo abnegado con el que se dedicaron y se lo considera extraordinario. Reconocer estas ideas instaladas cultural y socialmente nos puede ayudar a no reproducirlas.

[14] Ver texto completo de la ley en: https://www.argentina.gob.ar/normativa/nacional/ley-26150-121222

[15] Las puertas de entrada de la ESI son una herramienta de análisis y de reflexión para la implementación de la educación sexual integral en las instituciones educativas.

Afinar la mirada en cada momento de la clase

A la hora de pensar una clase desde la perspectiva de género y diversidad[16] tenemos que tener en cuenta que enseñamos en las dinámicas, las interacciones y las formas de comunicarnos que tienen lugar en el proceso de enseñanza y aprendizaje (llamado "currículo oculto o implícito"). El desafío es atender a estas dinámicas en cada momento de la clase y buscar herramientas creativas para desarmar esquemas que reproducen desigualdades y proponer relaciones sociales y educativas más inclusivas y transformadoras.

Al organizar grupos de trabajo

Durante el inicio de la clase, el momento de **organizar los grupos de trabajo** es una buena oportunidad para observar la clase desde una perspectiva de género y diversidad. Por ejemplo, cuando pedimos a las y los estudiantes que formen equipos, podemos preguntarnos: ¿logran formar grupos heterogéneos?, ¿trabajan siempre con sus amigas y amigos?, ¿quiénes "quedan afuera"?, ¿se separan en "chicos" y "chicas"?, ¿qué prejuicios sostienen estas formas de agrupamiento?

Las **dinámicas lúdicas** son herramientas para "romper el hielo" y promover la participación activa de todo el grupo. También son valiosas para facilitar el armado de grupos heterogéneos, motivando el intercambio con compañeras y compañeros con quienes no trabajen habitualmente. De este modo, se puede comenzar a promover el respeto por la diversidad y el aprendizaje en convivencia.

Estas dinámicas que promueven el armado de grupos diversos ayudan a movilizar los estereotipos en los roles sociales, generando experiencias educativas más inclusivas y enriquecedoras. Algunas **dinámicas lúdicas** son:

- Repartir al azar tarjetas con nombres de animales y que las y los estudiantes se agrupen con quienes tengan al mismo animal. Podemos aprovechar los personajes de Pilas Bloques (pingüino, yaguareté, ñandú, carpincho y picabuey) y sumar otros animales autóctonos.
- Repartir al azar tarjetas con "instrucciones" que cada estudiante debe realizar como si fuera un autómata ("levantar la mano derecha", "saltar sobre el pie izquierdo", "taparse la boca con ambas manos", "tararear una canción"). Las y los estudiantes deberán agruparse con quienes ejecuten la misma instrucción.
- Preparar tarjetas con partes de un rompecabezas. Se pueden usar los escenarios de los personajes de Pilas Bloques. Entregar una tarjeta a cada estudiante y pedirles que encuentren a quienes tengan las partes que encajen con la suya. Cada rompecabezas formará un grupo distinto.

[16] La perspectiva de género y diversidad brinda herramientas para visibilizar y transformar las desigualdades estructurales que afectan históricamente a mujeres y LGBTTI+ en el acceso y ejercicio de sus derechos.

Durante el trabajo en grupos

Ya durante el desarrollo de la actividad, el **recorrido por los grupos** es también una instancia clave para mirar cómo se vinculan las y los estudiantes dentro de cada grupo:

- ¿Trabajan de forma colaborativa?
- ¿Rotan el uso de la computadora?
- ¿Todas y todos acceden al desafío por igual o existe una división de tareas que deja a alguien fuera de la práctica?

Debemos atender a que estudiantes mujeres y de identidades de género trans y no binarias no sean relegadas al momento de operar o estar al frente de dispositivos computacionales. Afinar la mirada en este punto resulta un ejercicio muy valioso cuando tenemos en cuenta que las estudiantes y los estudiantes ingresan a la clase con un bagaje de saberes previos, muchas veces vinculados a los estereotipos de género, que pueden afianzar la brecha digital. En este sentido, la menor exposición que es posible que hayan tenido quienes no son varones durante la infancia al uso de las computadoras genera menor seguridad en su vínculo con las CC, especialmente en culturas en donde el acceso a juguetes y videojuegos están atravesados por prejuicios culturales de género[17]. Será importante incentivar la participación de todas y todos y rotar las tareas dentro del grupo, para dar la oportunidad a todas y todos de asumir diferentes roles en los distintos momentos de las actividades.

Durante los intercambios orales

En las secuencias, hacia el cierre de las actividades, proponemos un **intercambio oral**. En este momento, una tarea fundamental será cuidar y promover la distribución de la palabra. Desde el comienzo, podemos acordar con las y los estudiantes que las intervenciones deben ser respetuosas de la diversidad de posturas y opiniones. Una buena estrategia para impulsar el diálogo es formular preguntas: ¿alguien quiere aportar algo más?, ¿qué opinan las y los demás?, ¿alguien más ha pensado sobre lo dicho por sus compañeras y compañeros?, ¿qué piensa el resto? Una manera de "hacer fluir" la palabra puede ser rescatar el aporte de quienes no suelen participar y procurar que la palabra no sea monopolizada.

Tener paciencia y dar el tiempo necesario a quienes no suelen animarse a tomar la palabra también es importante. Es recomendable invitar a la participación evitando exponer a aquellas personas que les cueste más la palabra. Para ello, podemos habilitar formas de expresión alternativas al intercambio oral en voz alta. Como, por ejemplo, entregar a cada estudiante tarjetas para que escriban palabras claves que respondan a las preguntas disparadoras y las peguen en el pizarrón para ser luego retomadas en la conceptualización final.

[17] Echeveste, M. E., Gómez, M. J., Benotti, L. (2021). *Jornadas Argentinas de Didáctica de las Ciencias de la Computación.*

Al momento del cierre

Luego del intercambio, preguntamos por los conocimientos específicos, pero también indagamos las emociones y los sentimientos que emergieron durante la actividad disciplinar, con el uso de la computadora, durante el trabajo en grupo o en la división de tareas. Podemos preguntar: ¿cómo se sintieron haciendo la actividad?, ¿qué dificultades encontraron?, ¿pudieron resolverlas?, ¿cuáles sí y cuáles no?, ¿cómo se sintieron trabajando en grupo?, ¿cómo se organizaron?, ¿hay alguna tarea que les habría gustado hacer y se quedaron con las ganas de hacerlo? Si es así, ¿cómo se les ocurre que podríamos atenderlo para la próxima clase? Podrían anotarse las decisiones de mejora que se acuerdan para que estén visibles para las próximas clases y, como grupo, hacer un seguimiento.

En este tipo de intercambios, es posible que podamos detectar frustraciones y dificultades, así como también identificar si persisten ideas estereotipadas acerca de las habilidades que se esperan para estar atentas/os a abordarlas y desarmarlas en los siguientes encuentros.

Los personajes de Pilas Bloques y los desafíos que nos proponen

Las CC, al igual que el resto de las ciencias exactas, fueron narradas histórica y socialmente como disciplinas androcéntricas, donde los varones fueron los protagonistas[18]. Esto invisibilizó los valiosos aportes que mujeres y personas trans y de identidad de género no binaria hicieron y hacen a la disciplina, así como también, las barreras que debieron y deben enfrentar y las diversas formas de opresión que sufren y sufrieron.

Para propiciar experiencias educativas inclusivas y promotoras de la equidad de género y diversidades necesitamos revisar los contenidos específicos que forman parte de lo que enseñamos y también las imágenes que acompañan esos contenidos y que forman parte del currículo oculto o implícito.[19]

En este cuaderno, los autómatas que protagonizan los distintos desafíos fueron concebidos y diseñados con el objetivo de tensionar los estereotipos de género presentes en las CC y la programación habitualmente. En este sentido, buscan movilizar las ideas sociales en torno a los roles de género tradicionales, la identidad de género, las expectativas de comportamiento, las habilidades y las cualidades. Por ejemplo, los personajes femeninos, Yvoty y Mañic son adolescentes apasionadas por la ciencia y la tecnología. Capy y Guyrá rompen con el modelo de "varón tradicional" y se muestran cariñosos y cuidadores entre ellos y con su entorno natural.

Chuy es un personaje cuya identidad de género es no binaria, saliéndose del esquema binario varón-mujer, ofrece la posibilidad de generar identificaciones diversas.

---

[18] Ortmann, Cecilia (2019). "Clase virtual 3.G: La enseñanza de la ESI en el aula de Física, Química y Matemática". En *Módulo 3: Enseñanza de la ESI. Especificaciones por nivel y/o por área curricular.* Diplomatura de Extensión en Educación Sexual Integral. Facultad de Filosofía y Letras de la Universidad de Buenos Aires.

[19] *Ibidem.*

En las secuencias didácticas, aparecen estos personajes y, con ellos, las preguntas, los comentarios y las inquietudes de las y los estudiantes. Es recomendable aprovechar la oportunidad para intercambiar sobre estos temas, desarmando roles y estereotipos de género, promoviendo el respeto y la empatía hacia las otras y los otros y frenando la clase si irrumpen comentarios o situaciones violentas que ameriten una conversación más profunda. Asimismo, siempre podemos recurrir al trabajo en red y el intercambio con otras y otros docentes para abordar estos temas desde distintas perspectivas y de manera articulada de modo que se enriquezca el abordaje.

Secuencias didácticas

> Unidad 1:

Primitivas, procedimientos y repetición

1. **¿Qué es programar?** Autómatas, programas y primitivas
2. **Definimos nuestros bloques**. Procedimientos y repetición simple
3. **Programamos en papel cuadriculado**. Legibilidad y reutilización
4. **Programamos estrategias en Pilas Bloques**. Estrategia y división en subproblemas
5. **Creamos desafíos de repetición**. Repetición simple
6. **Seguimos programando estrategias en Pilas Bloques**. Estrategia y restricciones de orden
7. **Creamos desafíos de procedimientos**. Procedimientos

Secuencia didáctica 1

¿Qué es programar?

Autómatas, programas y primitivas

¿Cómo definiríamos qué es programar? ¿Qué tipo de instrucciones podemos darle a una computadora para que resuelva un problema? ¿Qué posibilidades y limitaciones tenemos cuando expresamos soluciones a problemas computacionales? ¿Qué es y cómo se usa Pilas Bloques?

En esta secuencia nos aproximamos a la elaboración de soluciones a problemas con instrucciones simples y precisas para comprender la existencia de restricciones al construir programas para computadoras. Trabajaremos en modalidad "desenchufada" con dramatizaciones y con ejercicios en papel para luego, presentar el entorno de programación Pilas Bloques.

Actividad 1

Las y los estudiantes le dan indicaciones a su docente para que alcance un objetivo como si se tratara de una computadora. De esta manera, comienzan a trabajar con estrategias de solución y se introducen nociones fundamentales de programación, como **problema computacional, autómata, programa, secuencia** y **primitiva**.

Actividad 2

Las y los estudiantes diseñan un problema para ser resuelto por un autómata, lo que les requiere poner en juego las nociones fundamentales de programación que surgieron en la actividad anterior.

Actividad 3

A partir del desafío en Pilas Bloques ***Capy y Guyrá***, las y los estudiantes exploran el entorno del programa, las nociones presentadas en las actividades anteriores, y se familiarizarán con la metodología de trabajo.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Soluciones a problemas computacionales

- Diseño de programas: estrategia.

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, secuencias.
- Ejecución secuencial de programas: ejecución, autómata.

Objetivos de aprendizaje

- Conocer el nivel de precisión y especificidad de los comandos primitivos empleados en programación y su relación con cómo deben formularse las soluciones computacionales.
- Aproximarse a las nociones de autómata, programa y primitiva o comando primitivo.
- Conocer el entorno de enseñanza de la programación Pilas Bloques y su metodología de trabajo.

Materiales necesarios

- Pizarra o pizarrón.
- Hojas de papel y útiles para escribir.
- Dispositivos con Pilas Bloques instalado o acceso a su versión en línea https://pilasbloques.program.ar/

Actividad 1
Todo al pie de la letra

Las y los estudiantes deben "programar" a su docente: le darán instrucciones para que realice una tarea como si fuera un robot o una computadora. De esta manera, se encontrarán con el desafío que implica formular una tarea con este tipo de instrucciones. Se enmarcará este trabajo con algunas nociones teóricas fundamentales.

Objetivos >

Se espera que las y los estudiantes:

- Expresen tareas en términos de expresiones acotadas, simples y formuladas de manera muy precisa.
- Se aproximen a las nociones de comando primitivo o primitiva, autómata y programa.

Inicio >

El **propósito de este momento** es introducir la tarea de programación a partir de las concepciones previas de las y los estudiantes sobre el tema.

Orientaciones

¿Qué programas de computadora conocen? ¿Para qué sirven? ¿Qué características tiene un programa? ¿Cómo y quiénes los crean? ¿Les parece que es una tarea fácil o difícil?

Buscamos relevar los saberes previos y los conocimientos intuitivos de las y los estudiantes sobre los programas de computadora y su creación. Esto nos dará herramientas para contextualizar esta secuencia como un primer paso en el aprendizaje de la programación. Sus respuestas también podrán ser insumo para contrastarlas con las reflexiones que surjan en el momento de cierre.

Desarrollo >

El **propósito de este momento** es introducir la noción de programa como una secuencia de instrucciones simples, formuladas de manera muy precisa y sin ambigüedades, que forman parte de un repertorio acotado (las primitivas) que una máquina (el autómata) puede llevar a cabo (ejecutar) automáticamente.

Orientaciones

Planteamos a las y los estudiantes que los docentes actuaremos como **autómatas**, es decir, como una máquina que (como las computadoras) sigue al pie de la letra las instrucciones que se le dan. Proponemos cumplir algún propósito o recorrido en el aula (por ejemplo, llegar a la puerta desde un rincón del aula y salir). El objetivo de las y los estudiantes es dar las instrucciones para cumplir el objetivo. Sin más explicaciones, comenzamos a escuchar instrucciones e iremos construyendo el programa en el pizarrón. Anotamos las instrucciones una debajo de la otra, aprovechando los errores que informa el autómata para descubrir de qué manera formular estas instrucciones.

A medida que las y los estudiantes realizan sus propuestas, pueden suceder las siguientes situaciones:

- **La instrucción podría ser incorrecta**, es decir que tal como está formulada, no forma parte del repertorio de instrucciones factibles o conocidas por el autómata. Esto puede deberse a la **complejidad** de la instrucción (podemos considerar que "ir hasta la puerta" o "avanzar 10 pasos" son instrucciones complejas mientras que "avanzar un paso" o "dar un paso adelante" son instrucciones lo suficientemente simples), a que la instrucción sea **ambigua** (si nos dicen "mover" podemos interpretarlo como moverse en el lugar o avanzar en cualquier dirección) o a que la instrucción esté **expresada de una manera sustancialmente diferente a la esperada** (por ejemplo, si esperamos "avanzar un paso" entonces "mover adelante" o "caminar" serán consideradas incorrectas). En estos casos, podemos responder "No sé hacer eso" para exponer las dificultades que implican resolver una tarea sólo con instrucciones simples y formuladas de manera muy precisa[1].
- **La instrucción puede ser correcta**, **pero no puede ser ejecutada** (por ejemplo, "abrir la puerta" si estamos lejos de la puerta o esta ya está abierta). En ese caso, indicamos que no podemos ejecutar la instrucción.
- **La instrucción sea correcta y pueda ser realizada**, **pero nos aleje de resolver el problema** (por ejemplo, caminar en el sentido contrario o avanzar sobre un banco o contra una pared). En ese caso, la ejecutamos, es decir, obedecemos la orden propuesta para mostrar que se aleja de lo buscado. Es probable que las y los estudiantes identifiquen esta situación y quieran corregirnos sobre la marcha o apelen a nues-

[1] Si surgieran comandos más avanzados, como la repetición o la alternativa condicional, los consideraremos como demasiado complejos dado que se trata de una actividad introductoria.

tro sentido común para que interrumpamos la ejecución. Esta es una oportunidad para reforzar que, como autómatas, no somos capaces de evaluar si las instrucciones nos ayudan a resolver el problema o no, solo ejecutamos las instrucciones recibidas en el orden en el que nos fueron dadas.

- **La instrucción puede estar correctamente formulada**, **puede ser ejecutada y nos permita resolver el problema**. En ese caso, seguimos la instrucción (la ejecutamos) y la registramos en el pizarrón con la precisión con la que debe ser formulada.

Cierre >

El **propósito de este momento** es conceptualizar la noción de *programa*, *comando primitivo* o *primitiva* y *autómata* y que las y los estudiantes reconozcan que el orden en el que se indican las primitivas en un programa es importante para que se cumpla el objetivo.

Orientaciones

Recuperamos con las y los estudiantes la experiencia del primer ejercicio para construir algunas definiciones presentes en cualquier problema de programación. Se puede recurrir a las concepciones previas por las y los estudiantes en el momento de inicio y hacer un refuerzo o contrastación con lo que comenten en el intercambio de esta instancia.

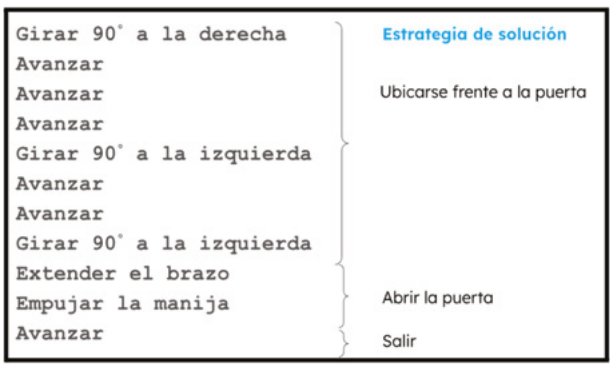
Es importante arribar a las siguientes nociones.

- Un **programa** es una descripción de una **solución** posible a un problema específico que se busca resolver con un **autómata** de forma automática, expresado como una **secuencia** de **instrucciones**. En esta secuencia, un ejemplo de programa sería lo que quedó escrito en el pizarrón al terminar el primer ejercicio.
- Las **instrucciones** con las que se construyen los programas deben ser lo suficientemente simples y estar formuladas de manera específica para que el autómata pueda reconocerlas y realizarlas, como señalamos cada vez que propusieron una instrucción que no sabíamos ejecutar. A estas instrucciones les diremos **comandos primitivos** o **primitivas**. Por ejemplo, una primitiva podría ser "avanzar un paso".
- Al proceso por el cual un autómata interpreta las instrucciones escritas en un programa y las lleva a cabo, le llamamos **ejecución** del programa.

¿Qué forma tiene el programa? ¿Se podrían intercambiar instrucciones de lugar y que el programa siga funcionando? ¿Cómo separarían el programa en partes? ¿Por qué?

Analizamos el programa escrito en el pizarrón para reflexionar sobre dos aspectos.

- Al observar la **forma secuencial del programa**, reflexionamos sobre el **orden** en el que aparecen las instrucciones en la secuencia: en algunos casos, será importante para que se resuelva el problema y en otros no. Podemos promover que las y los estudiantes especulen sobre cómo funcionaría el programa con las instrucciones en otro orden. Podemos señalar ejemplos en los que el orden sea importante, tales como en las situaciones en las que, para avanzar, primero se debe esquivar un obstáculo avanzando hacia un costado y luego avanzando por el camino libre; para el caso contrario, podemos identificar recorridos en terreno libre que involucren más de una dirección (por ejemplo, avanzar hacia el costado y hacia adelante sin obstáculos).
- Al observar el **programa completo**, podemos señalar que, si bien la solución está formulada como una única secuencia de instrucciones muy simples, podemos identificar **partes dentro de esta secuencia donde cada una resuelve una situación particular**. Estas situaciones son más complejas que las que puede resolver cada primitiva por separado pero más simple que el problema original. Estos son ***subproblemas***. Para ponerlo de manifiesto, proponemos que busquen estas partes en el programa y especifiquen qué resuelve cada una. Luego, leyendo la solución desde este enfoque, vemos que existe una **estrategia** para resolver el problema, que refleja la manera en la que elegimos resolverlo (por ejemplo, eligiendo un camino para llegar a la puerta) pero que no suele ser única (probablemente existan otros caminos que podríamos haber tomado). Resaltamos estas ideas para continuar trabajando con ellas en futuras actividades.



Un ejemplo de programa para salir del aula, en el que se utilizan primitivas de avance y de giros.

Actividad 2
Seamos autómatas

Profundizamos las ideas trabajadas en la actividad anterior cambiando los roles: serán las y los estudiantes quienes formulen los problemas para que sus compañeros y compañeras los resuelvan como si fueran autómatas.

Objetivos >

Se espera que las y los estudiantes:

- Demuestren comprensión de la noción de primitivas.
- Se aproximen a la noción de **problema computacional**.

Inicio >

El **propósito de este momento** es recuperar la noción de programa como solución a un problema y evidenciar los límites de los problemas que pueden ser resueltos por un autómata determinado.

Orientaciones

Dividimos la clase en grupos de dos o tres estudiantes. Podemos utilizar una dinámica lúdica para generar grupos al azar, con el objetivo de formar grupos heterogéneos.

Comenzamos planteando a las y los estudiantes que piensen un problema para ser resuelto por un autómata. Adelantamos que se lo entregarán a otro grupo para que lo resuelva con instrucciones paso a paso, del mismo modo en que lo hicimos en la **Actividad 1**, y, por eso, deberán describir claramente el problema e indicar todo lo que consideren necesario para que el otro grupo pueda construir una solución para que el autómata resuelva el problema.

¿Qué deberían tener en cuenta para definir un problema a ser resuelto por un autómata? ¿Qué información deberán pasarle al otro grupo para que pueda construir una solución al problema?

Estas preguntas pretenden que las y los estudiantes recuperen las nociones presentadas en la actividad anterior, con el objetivo de puntualizar la consigna de la actividad. Considerando esto, se puede orientar el intercambio hacia, por ejemplo, consensuar qué acciones podrán realizar los autómatas. Es importante hacer hincapié en que el relevamiento completo

de un problema requiere informar las primitivas, las condiciones iniciales y de contexto en las que se enmarca.

Podemos tomar nota en el pizarrón de las cuestiones que se acuerdan considerar al escribir el problema y la solución. De esta forma, podemos evidenciar que se trata de decisiones arbitrarias que se acuerdan para establecer un diálogo con el autómata.

Desarrollo >

El **propósito de este momento** es reforzar la comprensión de la noción de primitiva (una instrucción simple de un repertorio arbitrario y acotado, asociadas a un autómata y un problema en particular en un determinado contexto), y, sobre todo, las particularidades que conlleva formular programas a partir de ellas.

Orientaciones

En una primera parte, los grupos definen el problema que quieren que el otro grupo resuelva. Acompañamos el trabajo de los grupos para que logren una versión escrita del problema a resolver, prestando atención a que:

- incluya el autómata y las primitivas disponibles.
- pueda resolverse únicamente con las primitivas planteadas y con una solución no demasiado extensa.

Para que puedan verificar esto, podemos proponer que se coloquen en el lugar de quién recibirá el problema y ensayen una solución modelo.

En una segunda parte, los grupos intercambian sus problemas para resolverlos. Atendemos al intercambio entre los grupos por si no se comprende qué hace una primitiva o alguna pauta del problema a resolver. Asimismo, podemos pedirles que registren las dificultades que encuentren durante la elaboración y la resolución del problema para discutirlas en el momento de cierre, especialmente, si las primitivas no eran suficientes, no era claro su funcionamiento o eran muy complicadas de ejecutar.

Para desarrollar la solución, deben escribir claramente las instrucciones en orden secuencial y cumplir con los acuerdos que anotamos en el pizarrón o pizarra en el inicio de esta actividad.

Para probar o presentar la solución, pueden actuar físicamente en el aula (como hicimos en la actividad anterior) o simularla en un esquema en papel.

Cierre >

El **propósito de este momento** es identificar cómo las particularidades del trabajo con primitivas impactan en las maneras de pensar la solución y en el tipo de problemas que podemos resolver, y así comenzar a construir la noción de *problema computacional*.

Orientaciones

¿Qué tan fácil o difícil fue comprender cuál era el problema planteado por el otro grupo? ¿Pudieron resolver el problema con las primitivas disponibles? ¿Por qué? ¿Había alguna primitiva que fuera muy general o que se pudiera haber separado en acciones más chicas? ¿Hubo algún aspecto del problema que no pudiera resolverse?

Recuperar las dificultades encontradas en la solución de los problemas planteados es una oportunidad para revisar y reforzar las nociones abordadas en la actividad anterior (programa, autómata y primitivas) y sus particularidades (el autómata puede ejecutar un repertorio acotado de primitivas; y estas tienen que estar formuladas de manera muy precisa, porque el autómata ejecuta las instrucciones "sin pensar" ni tomar decisiones sobre ellas).

También nos interesa señalar que, así como las especificidades del trabajo con autómatas y primitivas impacta en la forma de las soluciones que construimos, también influye en el tipo de problemas que podemos resolver, como vieron cuando tuvieron que pensar el problema para las y los compañeros. En este sentido, a los problemas que podemos resolver con un autómata (en particular, con una computadora), les diremos **problemas computacionales**. No nos interesa ensayar una definición cerrada pero sí identificar que al resolver problemas con autómatas y primitivas hay restricciones y tenemos que conocerlas al construir soluciones (que posiblemente sean diferentes a las que conocemos de la vida cotidiana).

Material de referencia para docentes

¿Por qué decidimos no hablar de algoritmos (todavía)?

En el ámbito de las Ciencias de la Computación, un **algoritmo** es una descripción general de un único proceso que permite construir una solución para todas las instancias de un problema, por ejemplo, la descripción de un método que permita transformar una lista cualquiera de números para que quede ordenada de menor a mayor, en el que podemos identificar claramente el modelo de la computación como "entrada, transformación y salida de información". Asociar a los algoritmos con una secuencia de pasos para resolver tareas cotidianas esconde con su simplicidad aspectos que son inherentes a la noción disciplinar, construyendo una definición errónea de un término clave para la disciplina y poco relevante para las y los estudiantes (dado que, en este nivel, solo se usaría para nombrar un orden de pasos). Abordar esta noción central de la Computación requiere hacerlo en una instancia posterior de la formación de las y los estudiantes en la que puedan razonar en general y en abstracto.

Arribamos a esta decisión a partir de discusiones, experiencias e investigaciones en relación con el término y las dificultades que presentó su uso simplificado o banalizado.

Más información al respecto en: Dabbah, J., *et al. Propuesta curricular para la inclusión de las Ciencias de la Computación en la educación obligatoria de Argentina*. Fundación Sadosky, 2024. Disponible en línea. https://curriculum.program.ar/wp-content/uploads/2023/01/Program.ar_Propuesta-Curricular-para-la-inclusion-de-las-Ciencias-de-la-Computacion.pdf

Actividad 3
Capy y Guyrá.
Conocemos Pilas Bloques

Las y los estudiantes abordan el desafío ***Capy y Guyrá***, que motiva a explorar el entorno de programación Pilas Bloques aplicando las nociones fundamentales de programación presentadas en las actividades anteriores.

Objetivos >

Se espera que las y los estudiantes:

- Conozcan el entorno de programación Pilas Bloques y su dinámica de trabajo.

Inicio >

El **propósito de este momento** es motivar dinámicas de trabajo que habiliten a las y los estudiantes a explorar Pilas Bloques, prestando atención y reforzando que no haya desigualdades en el uso y el acceso a los dispositivos y en el tipo de tareas que realice cada integrante del grupo.

Orientaciones

Esta es una nueva oportunidad para relevar la experiencia de las y los estudiantes en el uso de computadoras, tecnologías digitales, videojuegos y programación. Esta información nos será muy útil para identificar quiénes cuentan con más y con menos experiencia, y procurar aprovechar esa diversidad para beneficio de todos favoreciendo el intercambio en grupos heterogéneos.[2]

Se propone que las y los estudiantes formen pequeños grupos heterogéneos, idealmente de a dos, e ingresen al desafío ***Capy y Guyrá*** en Pilas Bloques, del nivel intermedio.

Dado que es probable de que sea la primera vez que acceden a los desafíos de Pilas Bloques es importante acompañar el acceso al desafío propuesto.

[2] Para ello, una actividad sencilla, es indicar que levanten la mano, golpeen la mesa o den un aplauso quienes *"tengan computadora en su casa", "usen internet todos los días", "jueguen videojuegos", "alguna vez programaron"*, etc.

Es conveniente explicitar la dinámica de trabajo en grupos, prestando atención a explorar el entorno en conjunto, discutir y consensuar las decisiones que tomarán para resolver el desafío y acceder al dispositivo para asegurar que todas y todos puedan trabajar en Pilas Bloques de forma equitativa en cuanto a tiempo de uso y tipo de tareas.

Desarrollo >

El **propósito de este momento** es que las y los estudiantes exploren el entorno de programación Pilas Bloques para comenzar a construir la dinámica de trabajo en la plataforma.

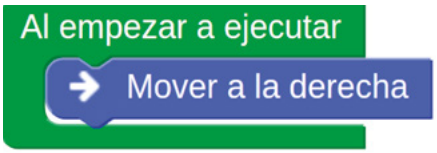
Orientaciones

La dinámica de este momento está fuertemente orientada hacia la **exploración de la herramienta por parte de las y los estudiantes**. Como docentes, nuestro rol será alentar esta exploración y resolver inquietudes para sostener este proceso evitando frustraciones.

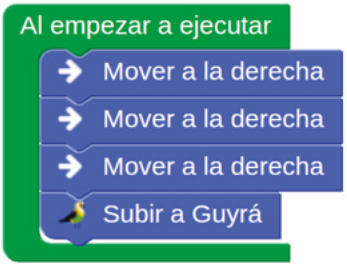
Mientras que las y los estudiantes resuelven el desafío, y en la medida que sea necesario, se puede orientar la exploración mediante la señalización de elementos presentes en el desafío que representan alguna de las nociones fundamentales de programación que se vienen trabajando desde el inicio de la secuencia. En este sentido, y después del proceso de exploración o incluso después de que algunos grupos hayan resuelto el problema, se propone identificar: la consigna y la pista del desafío, el escenario o contexto (y sus elementos), el espacio para armar el programa y el menú de bloques que se despliega a partir de la categoría *Primitivas*.

En cuanto al manejo de los dispositivos, es probable que haya quienes necesiten ayuda adicional (de nuestra parte o de sus compañeros y compañeras) en función de su experiencia previa, tal como podemos haberlo identificado al comienzo de la clase. Es importante reforzar el acompañamiento en estos casos para no profundizar desigualdades en el acceso y uso a la tecnología.

Considerando que este es el primer desafío en Pilas Bloques, es probable que requieran asistencia en la elaboración del programa, en particular, en determinar que el programa se construye arrastrando y encastrando bloques según sus formas. En este sentido, se les puede orientar a observar que el bloque Al empezar a ejecutar tiene un pico hacia abajo y las primitivas Mover a la derecha y Subir a Guyrá tienen esa misma forma, que encastraría como si fuera un rompecabezas.

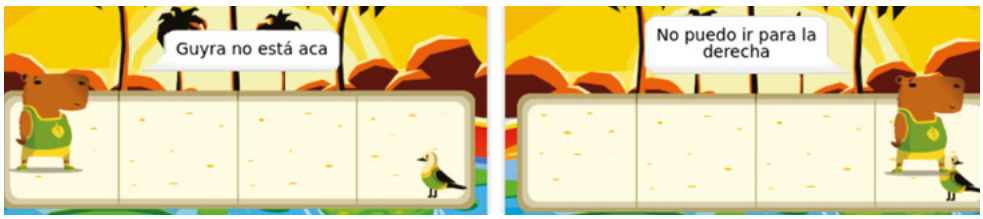


Invitamos a aquellas y aquellos estudiantes que hayan completado el programa, pero no lo hayan ejecutado aún, a que lo hagan para asegurarse de que funciona correctamente. Recordamos que el autómata resuelve el problema **ejecutando** el programa y, para tal fin, tienen disponible el botón ▶ EJECUTAR. Les indicamos que lo presionen y observen cómo, a medida que se ejecuta el programa, se resalta el bloque que está siendo ejecutado.



Un programa que resuelve el desafío.

Los personajes, en su rol de autómatas, informarán si ocurren errores durante la ejecución del programa (por ejemplo, ejecutar la primitiva Subir a Guyrá fuera del casillero en el que se encuentra o intentar desplazar el personaje fuera del tablero).



Ejemplos de errores durante la ejecución del programa.

Si no ocurrieron errores y se cumplió el objetivo del desafío, aparecerá un cartel que indica que pudieron resolver el desafío.

Cierre >

El **propósito de este momento** es comenzar a ensayar una dinámica para el trabajo con Pilas Bloques, relacionándola con las nociones de programación trabajadas en la secuencia.

Orientaciones

¿Cómo se organizaron para explorar el entorno de Pilas Bloques? ¿Lo hicieron en conjunto? ¿Por turnos? ¿Se repartieron las tareas? ¿Pudieron ponerse de acuerdo para decidir las instrucciones para crear el programa? ¿Qué dificultades encontraron en el trabajo en grupo? ¿Cómo las resolvieron?

En este cierre, aprovechamos para retomar los acuerdos que hicimos en el inicio de la actividad sobre la dinámica de trabajo en grupos[3].

¿Cómo supieron qué había que hacer, es decir, cuál era el problema a resolver? ¿Cómo supieron qué primitivas podía ejecutar el autómata? ¿Dónde las encontraron? ¿Cómo supieron si el programa funcionaba? ¿Cómo se ejecuta un programa en Pilas Bloques?

En este intercambio nos interesa recuperar de la experiencia:

- cuestiones **instrumentales** fundamentales para trabajar en **Pilas Bloques**, como la ubicación de cada uno de los elementos de un desafío como la consigna, la pista y el escenario, las primitivas y la forma de crear un programa (o solución) y cómo ejecutarlo. Podemos hacer una revisión del entorno para ir identificando dónde aparecen y cómo accedemos a cada uno de ellos.
- la idea de que **en el trabajo en computadora están presentes las mismas nociones** que vimos en las actividades anteriores de la secuencia, para asociar a los personajes con el autómata y a los bloques azules con las primitivas.

[3] Podemos aprovechar la ocasión para indagar qué sucedió en torno al uso compartido de la computadora y la división de tareas. Podemos promover la reflexión sobre si se manifestaron ideas estereotipadas acerca de las habilidades asociadas al género u otros prejuicios o preconceptos y movilizar cambios al respecto acompañando al grupo.

Secuencia didáctica 2

Definimos nuestros bloques

Procedimientos y repetición simple

¿Es lo mismo un problema largo que un problema complejo? Cuando una acción debe repetirse en un programa, ¿podemos evitar agregar muchas veces los mismos bloques? ¿Es posible crear nuestros propios bloques?

En esta secuencia se presentan los **procedimientos**, una herramienta de programación que permite reutilizar soluciones a subproblemas y definir acciones propias. También se aborda la **repetición simple** como un comando que facilita la tarea de elaboración e interpretación de programas.

Actividad 1

A partir del desafío de Pilas Bloques ***Nuevos Comandos***, las y los estudiantes se aproximan al uso de procedimientos como herramienta de reutilización de soluciones al tener que resolver un subproblema presente más de una vez en el problema general.

Actividad 2

A partir del desafío de Pilas Bloques ***Chuy hace jueguito***, las y los estudiantes utilizan procedimientos para explicitar la estrategia de solución en términos del dominio del problema.

Actividad 3

A partir del desafío de Pilas Bloques ***No me canso de rebotar***, las y los estudiantes se aproximan al uso de la repetición simple.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Soluciones a problemas computacionales

- Diseño de programas: estrategia, legibilidad.

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: procedimientos, repetición.

Objetivos de aprendizaje

- Proponer soluciones que empleen procedimientos que reutilicen otras soluciones que fueran creadas para resolver subproblemas.
- Proponer soluciones que empleen procedimientos motivados por la definición de acciones relacionadas con el dominio del problema para lograr un correlato entre la estrategia de solución y el problema.
- Proponer soluciones que aprovechen la repetición simple para ejecutar un comando una cantidad determinada de veces.

Saberes previos de CC

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, secuencias.
- Ejecución secuencial de programas: ejecución, autómata.

Materiales necesarios

- Dispositivos con Pilas Bloques instalado o acceso a su versión en línea https://pilasbloques.program.ar/

Actividad 1
Nuevos comandos

Presentamos los procedimientos como una herramienta de programación que permite reutilizar soluciones de subproblemas que se repiten dentro de un problema.

Objetivos >

Se espera que las y los estudiantes:

- Asocien el uso de procedimientos con la reutilización de soluciones de subproblemas y los definan e invoquen en el programa solución.
- Reconozcan la importancia de una denominación representativa de los procedimientos para facilitar la comprensión de la estrategia de solución.

Inicio >

El **propósito de este momento** es reforzar los elementos básicos para la creación de un programa en Pilas Bloques a partir de los comandos disponibles[1] en el desafío y familiarizarse con el entorno.

Orientaciones

Dividimos la clase en **grupos heterogéneos** de dos o tres estudiantes[2].

Una vez formados los grupos, las y los estudiantes ingresan al desafío ***Nuevos comandos*** en Pilas Bloques. Dado que este desafío es de los primeros que resuelven en Pilas Bloques, es conveniente recorrer los grupos para observar los avances y las dificultades, especialmente, en cuanto a la exploración del entorno y la creación de programas. Podemos aprovechar para reforzar la **noción de ejecución** y la importancia del orden de las **primitivas**, además del manejo básico de los bloques y la información disponible en el entorno (consigna, pistas, etcétera).

Cuando los grupos hayan conseguido que Capy y Guyrá recojan el primer conjunto de latas de la derecha y estén acercándose al inicio de la solu-

[1] En la secuencia "¿Qué es programar?" se abordan las nociones de programa y primitiva y se propone una primera actividad de programación en Pilas Bloques.
[2] Podemos utilizar una dinámica lúdica, por ejemplo, repartir al azar tarjetas con nombres de animales (como los de Pilas Bloques: pingüino, yaguareté, ñandú, carpincho y picabuey) y que las y los estudiantes se agrupen con quienes tengan al mismo animal.

ción del segundo conjunto de latas y todas y todos hayan podido explorar la herramienta, podremos avanzar al siguiente momento.

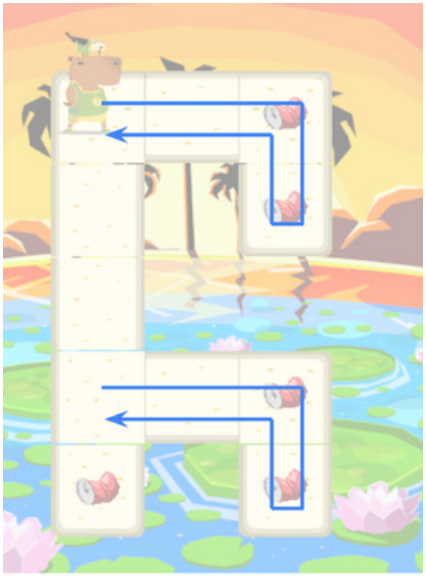
Desarrollo >

El **propósito de este momento** es presentar los procedimientos como una herramienta de programación que permite definir una solución a un subproblema y que puede ser reutilizada en una solución general cada vez que el mismo subproblema aparezca.

Orientaciones

El **disparador de este momento** es que las y los estudiantes identifiquen, a partir de la observación del escenario, que para completar el desafío tienen que resolver un problema idéntico al que resolvieron en la primera parte.

*Comparen la parte superior del escenario, que ya resolvieron, con la inferior, que les queda por resolver. ¿Será posible aprovechar lo que ya resolvieron para resolver lo que falta? ¿Alguien utilizó o intentó utilizar **procedimientos**, el nuevo bloque disponible en este desafío? Si lo utilizaron, ¿qué pudieron probar? ¿Qué inconvenientes encontraron? ¿Para qué creen que sirven los procedimientos?*



Invitamos a que las y los estudiantes que hayan experimentado con procedimientos relaten las pruebas realizadas, los descubrimientos y los inconvenientes encontrados en cuanto a uso y propósito de esta herra-

mienta de programación. La recuperación de estas experiencias permite enmarcar y motivar los siguientes momentos.

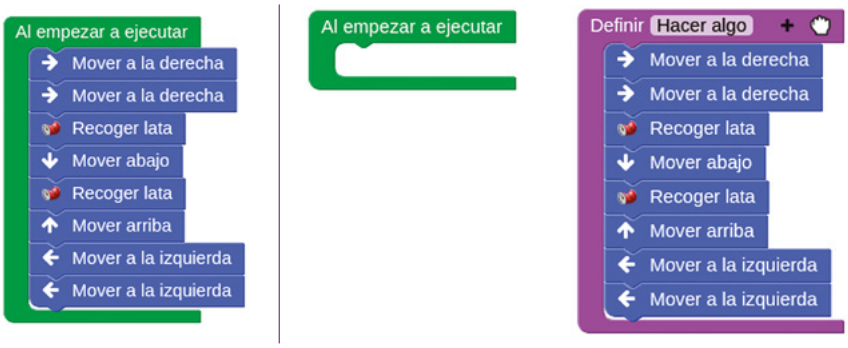
Material de referencia para docentes

Algunos términos clave

- El término **bloque** se refiere a cada elemento disponible en Pilas Bloques para crear un programa. No solo las primitivas son bloques; también lo son las repeticiones, las alternativas condicionales, las definiciones de procedimientos e incluso el programa principal.
- El término **comando** se refiere a bloques que representan acciones y tienen efectos directos en el entorno, el escenario o la ejecución del programa. Las primitivas, las repeticiones y las alternativas condicionales son ejemplos de comandos. En Pilas Bloques, su representación gráfica incluye picos que indican un encastre en secuencia o tienen espacio para el encastre de otros comandos en su interior (como veremos más adelante con la repetición simple).
- Al hablar de **primitivas** nos referimos a las acciones que el autómata (el personaje) puede realizar "por defecto", representadas con los bloques azules que vienen predefinidos en el desafío, por ejemplo, en el desafío trabajado, los bloques Mover a la derecha, Mover arriba, Recoger lata. Las primitivas dependen del autómata y el entorno; son unas de las piezas básicas con las que **construimos los programas**.

A continuación del intercambio, realizamos una demostración de la creación, definición e invocación de un procedimiento para la primera parte del desafío. Para crearlo, deben arrastrar un bloque Definir al espacio de trabajo. Luego, deben sacar del programa principal los bloques de las primitivas que resuelven la primera parte del desafío y encastrarlas dentro del procedimiento. Así queda definido el procedimiento.

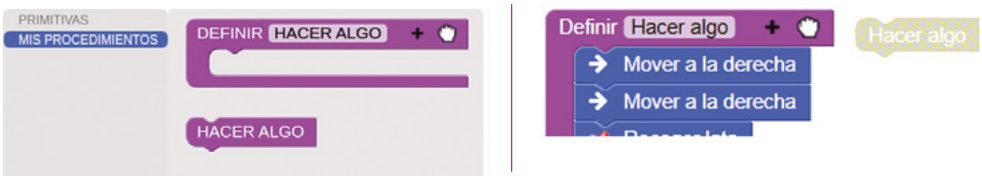
Para quienes tengan dificultades sobre dónde deberían ubicarse estas primitivas, podemos identificar similitudes en la forma de los bloques de definición de los procedimientos y de construcción del programa principal.



A la izquierda, la solución al problema de recoger las latas de la derecha dentro del bloque principal. A la derecha, la solución al problema pero trasladada al procedimiento.

Cuando todos los grupos hayan definido el procedimiento, les proponemos continuar con la solución del desafío. En esta instancia es importante observar si las y los estudiantes se encontraron con la necesidad de **invocar** el procedimiento, pero no supieron cómo hacerlo.

Para motivar la invocación del procedimiento en aquellos grupos que no avanzaron en ese sentido, podemos proponer ejecutar el programa para que vean que Capy y Guyrá no realizan las acciones de las primitivas que movieron para definir el procedimiento. En un caso u otro, para que encuentren el bloque de **uso o invocación** del procedimiento, proponemos revisar *Mis procedimientos* y/o ver qué sucede si presionan el icono de la mano blanca ( ) en la parte derecha del bloque de definición. Para explorar el funcionamiento de estos bloques, invitamos a buscar similitudes con la forma de los bloques que representan las primitivas. También podemos alentar que invoquen el procedimiento más de una vez, retomando la intención inicial de aprovechar la solución al primer subproblema para completar el desafío.



Hay dos maneras para acceder al bloque que nos permite usar o invocar un procedimiento en el programa. Como se ve en la captura de la izquierda, los bloques comando aparecen en la categoría "Mis Procedimientos". En la captuera de la derecha, vemos que, al hacer clic sobre el icono de la mano, se genera un bloque. Este se verá grisado hasta que esté efectivamente usado dentro del programa.

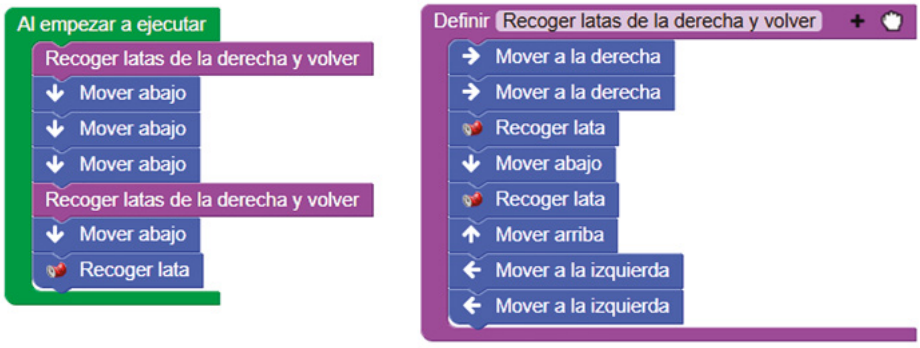
A medida que vayan avanzando con la solución del desafío, es posible que haya que recordar **algunas pautas en el uso de procedimientos** que surgieron en la demostración.

Podemos proponer una puesta en común para que compartan su experiencia en el uso de procedimientos. Con estas ideas en circulación, orien-

tamos a las y los estudiantes a que realicen las adaptaciones necesarias para completar la solución utilizando un mismo procedimiento para resolver las dos apariciones del subproblema.

Otro punto a tener en cuenta (aunque lo enfatizaremos en los desafíos siguientes[3]) es la importancia de asignarle una **denominación representativa** al procedimiento. Es decir, aprovechar la posibilidad de completar el **bloque de definición del procedimiento** con un texto que refleje qué hace efectivamente el procedimiento, por ejemplo, Recoger latas de la derecha y volver. Esta etiqueta o nombre del procedimiento aparecerá en los **bloques de invocación**.

Si otra persona que no conoce el escenario viera el programa principal, ¿creen que entendería qué hace el bloque Hacer algo? ¿Qué hace el bloque Mover abajo? ¿Cómo lo sabemos? ¿Qué podríamos hacer para que se entienda qué hace el procedimiento?



Cierre >

El **propósito de este momento** es recuperar las nociones claves sobre procedimientos y conceptualizarlos como una manera de crear bloques y reutilizar soluciones a subproblemas que se repiten dentro de un problema.

Orientaciones

Habilitamos un espacio para promover un **intercambio** en el que todas y todos puedan expresarse y compartir algo de su experiencia, podemos motivar diversas formas de expresión alternativas al intercambio oral en voz alta. El objetivo del intercambio es repasar:

- qué pasos son clave en el trabajo con procedimientos y cómo se resuelve cada uno (la definición, la invocación y la elección de un nombre o una denominación);

[3] Ver el índice de la colección.

- que el trabajo con procedimientos permitió reutilizar una parte de la solución y esto nos ahorró tener que volver a resolver un problema que teníamos una solución para él.

¿Se imaginan cómo habría sido resolver el desafío si hubieran contado con la primitiva Recoger latas de la derecha y volver? ¿Qué ventajas nos brinda poder definir nuestros propios procedimientos?

¿Cómo se sintieron haciendo la actividad? ¿Qué dificultades encontraron? ¿Cómo las resolvieron? ¿Pudieron resolver todas las actividades? ¿Cómo se sintieron trabajando en grupo? ¿Pudieron participar todas y todos? ¿Hay alguna tarea que les habría gustado hacer y no pudieron? Si sucedió, ¿cómo podríamos evitarlo en la próxima clase? [4]

Como una conceptualización más profunda, nos interesa presentar al uso de procedimientos como una manera de ampliar las primitivas disponibles según nuestras propias necesidades. Por ejemplo, en este caso, podemos pensar que una vez definido el procedimiento Recoger latas de la derecha y volver, agregamos al desafío una primitiva para recoger las latas y, en este nuevo entorno, construimos la solución completa. En este sentido, **definir procedimientos es una manera de facilitar el proceso de programación,** pues nos permite resolver un problema contando con comandos armados por nosotras y nosotros específicamente para un subproblema que nos interese.

[4] En este cierre, preguntamos por los conocimientos específicos, pero también indagamos las emociones y los sentimientos que se movieron con el trabajo en grupos, con el uso compartido de la computadora y la división de tareas. Es una ocasión en la que podemos detectar frustraciones y dificultades, y rastrear si se manifestaron ideas estereotipadas acerca de las habilidades asociadas al género u otros conceptos que hayan causado inconvenientes, para que podamos estar atentos en las siguientes actividades.

Actividad 2
Chuy hace jueguito

En esta actividad se trabajará con un desafío de Pilas Bloques donde el autómata debe realizar una secuencia ordenada de acciones, que no pueden resolverse con una única primitiva.

Objetivos >

Se espera que las y los estudiantes:

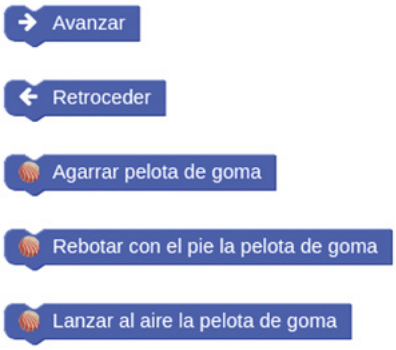
- Definan procedimientos para crear acciones más complejas que las expresadas en las primitivas.
- Utilicen la denominación de los procedimientos para expresar claramente la estrategia de solución en el programa.

Inicio >

El **propósito de este momento** es presentar la consigna del desafío y analizarla para descubrir sus partes y así poder dividir el problema en una serie de subproblemas, que sean más fáciles de resolver.

Orientaciones

Indicamos a las y los estudiantes que ingresen al desafío ***Chuy hace jueguito*** de Pilas Bloques, y les pedimos que lean y comenten la consigna y la pista. Además de recuperar cuestiones instrumentales relativas al uso de la plataforma, alentamos a que identifiquen diferentes partes en las que el objetivo de la consigna puede ser dividido (*Avanzar, Calentar, Hacer jueguito, Volver a su lugar*), comparando con las primitivas disponibles del desafío. De esta forma, se espera que adviertan que no hay bloques disponibles para *Hacer jueguito* ni *Calentar*.



Primitivas disponibles en el desafío.

Desarrollo >

El **propósito de este momento** es que las y los estudiantes resuelvan un problema cuya estrategia requiere de la definición de procedimientos para crear "bloques nuevos" que resuelvan acciones más complejas que las primitivas disponibles.

Orientaciones

Se propone a las y los estudiantes que comiencen a resolver el desafío.

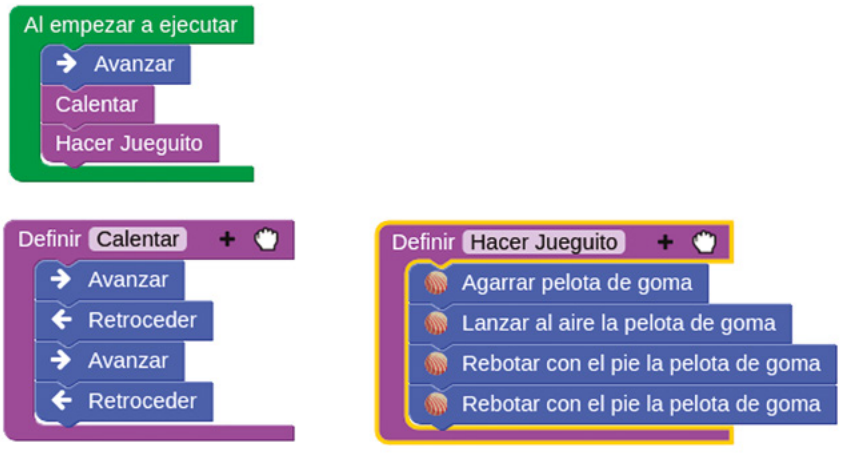
Es importante prestar especial atención a la dinámica entre estudiantes dentro de cada grupo. Incentivamos la participación de todas y todos en el manejo de la herramienta. Animamos y acompañamos a las y los estudiantes que presenten más dificultad.

Invitamos a los grupos a comparar las estrategias que propusieron con las primitivas disponibles.

Podemos recuperar lo trabajado en la **Actividad 1** sobre procedimientos y cómo **crear nuevos bloques que reflejen acciones que no están disponibles como primitivas**.

Luego, alentamos y acompañamos a los grupos para que los creen (por ejemplo, un procedimiento llamado Calentar que resuelva esta parte de la consigna).

Una situación análoga sucede con la acción Hacer jueguito, ya que no aparece una primitiva para resolverlo.



Solución incompleta al desafío, que resuelve las acciones "Calentar" y "Hacer jueguito" con procedimientos.

Resta resolver cómo lograr que Chuy logre volver a su lugar. Esto puede hacerse con el agregado de un bloque Retroceder, pero también puede ser que aparezca una solución que implemente un procedimiento Volver a su lugar que comprenda el comando Retroceder. Esta solución es válida

y pone en evidencia la diferencia entre una primitiva (Retroceder) y una acción "Volver a su lugar", en el sentido de que una primitiva hace referencia a una instrucción que el autómata puede llevar a cabo, mientras que una acción hace referencia al problema que se busca resolver.

La diferencia entre primitiva y acción es sutil y se refuerza en otras secuencias de la colección como parte del trabajo sobre las estrategias y la importancia de la legibilidad de los programas escritos.

En el ejemplo, para lograr que Chuy vuelva a su lugar (acción) tenemos que utilizar la primitiva Retroceder; para representar acciones en nuestros programas podemos construir bloques con significado específico y, para eso, utilizamos procedimientos con denominaciones que hagan referencia a las acciones que representan.

Dos soluciones al desafío. A la derecha, se define un procedimiento para la acción "Volver a su lugar".

Cierre >

El **propósito de este momento** es reflexionar sobre la construcción de procedimientos para representar las acciones del problema a resolver en el programa con las primitivas disponibles.

Orientaciones

¿Pudieron distribuir las tareas de otro modo en esta nueva actividad? ¿En esta experiencia detectaron algo diferente en la forma de resolver dificultades? ¿Cómo se sintieron?

Buscamos promover con un intercambio oral una reflexión que permita a las y los estudiantes introducir modificaciones en el modo de trabajo en grupo y en los intercambios que les permitan atender a las emociones e inquietudes de sus pares.

¿Definieron nuevos procedimientos? ¿Cuántos? ¿Qué problema resuelve cada uno de ellos?

De manera similar a la actividad anterior, este desafío nos presenta la oportunidad de usar procedimientos para ampliar las primitivas disponibles para resolver el problema. En el ejemplo, como no existían comandos que hicieran que Chuy "haga un calentamiento" y "haga jueguito" fue conveniente definir procedimientos que representen esas acciones. De esta manera, se facilita la tarea de programación, pues contamos con nuevos bloques para resolver subproblemas específicos dentro del problema general.

¿El programa que crearon resuelve todos los subproblemas presentes en el desafío? ¿Cómo pueden asegurarlo?

Estas preguntas proponen que las y los estudiantes identifiquen si hay un procedimiento o primitiva por cada una de las acciones del desafío: "Avanzar", "Calentar", "Hacer jueguito", "Volver a su lugar". Además propone que analicen las denominaciones que le dieron a los procedimientos para verificar que hacen referencia a las tareas que Chuy debe realizar para resolver el problema. Esto es deseable, pues facilita la comprensión de la estrategia de solución y, por lo tanto, la programación en equipos de trabajo.

Actividad 3
No me canso de rebotar

En esta actividad presentamos la **repetición simple** como una herramienta de programación que permite repetir la ejecución de un mismo comando una cantidad determinada de veces.

Objetivos >

Se espera que las y los estudiantes:

- Conozcan el bloque Repetir y lo utilicen para ejecutar un comando una cantidad determinada de veces.
- Reconozcan la conveniencia de utilizar repeticiones para facilitar la comprensión de la estrategia de solución para problemas repetitivos.

Inicio >

El **propósito de este momento** es motivar la exploración autónoma del entorno del desafío de Pilas Bloques ***No me canso de rebotar*** para elaborar una estrategia de solución basada en los comandos disponibles, siguiendo la estrategia didáctica de aprendizaje por indagación.

Orientaciones

Siguiendo la propuesta didáctica[5], proponemos que ingresen al desafío para **explorar e intentar resolverlo**.

Aprovechamos para recorrer los grupos y registrar las dinámicas entre sus integrantes. Incentivamos la participación de todas y todos en el manejo de la herramienta. Animamos y acompañamos a quienes encuentren más dificultad.

[5] Se pueden encontrar propuestas de dinámicas lúdicas en "Enseñar computación desde la mirada de la Educación Sexual Integral (ESI)", 11.

Desarrollo >

El **propósito de este momento** es introducir la noción de repetición a partir de la experiencia con el desafío de Pilas Bloques.

Orientaciones

Durante la exploración del entorno, las y los estudiantes pueden descubrir que la consigna es hacer que Chuy rebote la pelota de ping pong 30 veces y que, para cumplir esta tarea, disponen solamente de la primitiva Rebotar pelota de ping pong.

Es probable que la primera solución que ensayen sea colocar muchas veces el bloque. Cuando surja la inquietud de que es una tarea repetitiva y demasiado tediosa para un problema simple, podemos motivar la exploración del bloque Repetir. Cuando creamos pertinente, interrumpimos el trabajo de los grupos para hacer una puesta en común sobre cómo se usa y para qué sirve este bloque.

Cuando las y los estudiantes hayan finalizado, realizamos una segunda puesta en común para que puedan socializar las soluciones elaboradas y, en particular, contar cómo utilizaron el bloque Repetir.

Cierre >

El **propósito de este momento** es conceptualizar y valorar el uso de la repetición simple cuando se deba ejecutar un mismo comando varias veces para agilizar la tarea de programación y facilitar la comprensión de la estrategia de solución.

Orientaciones

En este cierre de la secuencia, para habilitar **formas de expresión alternativa** al intercambio oral, podemos proponerles escribir una reflexión personal sobre lo aprendido.

¿Qué aprendieron del trabajo con sus pares a lo largo de las actividades? Partiendo de su experiencia, ¿podrían escribir tres consejos para un buen trabajo en equipo?

En este momento de la reflexión, buscamos retomar algunas de las dificultades que hayan compartido o hayamos registrado durante los intercambios en las **Actividades 1** y **2**. Podemos pedirles que lean en voz alta los consejos o invitarlos a anotarlos y compartirlos en un mural.

¿Cómo sería el programa que resuelve este desafío si no hubieran utilizado la repetición simple? ¿Podríamos darnos cuenta fácil si faltan o sobran bloques con solamente mirar el programa? ¿Sería igual de fácil o difícil resolver el desafío si Chuy tuviera que rebotar la pelota 50 veces o 100 veces?

Estas preguntas pretenden que las y los estudiantes reflexionen acerca de las ventajas de utilizar repeticiones y priorizar el uso de esta herramienta por cuestiones de facilidad de comprensión de la solución. En particular, se propone abordar estas reflexiones mediante la comparación con la solución del desafío si no se hubiera utilizado repetición simple. En este sentido, las y los estudiantes pueden identificar la conveniencia de utilizar esta herramienta de programación por la facilidad en la corrección o modificación del programa si hubiera que realizar más o menos repeticiones, pues basta con modificar el valor que determina la cantidad de repeticiones en el bloque Repetir.

¿Para qué se usa el bloque Repetir en los programas? ¿Qué otros problemas se imaginan que se podrían resolver utilizando este bloque? ¿En qué artefactos o programas que conozcan les parece que se usan comandos similares al bloque Repetir?

Como instancia final de la reflexión, propiciamos una conceptualización y una generalización de esta herramienta. Nos interesa caracterizar a este bloque como un comando que se utiliza para indicar que otros comandos deben ejecutarse una determinada cantidad de veces; en este sentido, propiciamos la búsqueda de otros casos donde la repetición podría ser de utilidad. Por ejemplo, el programa que controla una impresora puede usar un bloque Repetir si debe imprimir muchas veces un mismo documento.

Secuencia didáctica 3

Programamos en papel cuadriculado

Legibilidad y reutilización

¿Podemos dividir un problema complejo en partes y resolverlas por separado? ¿Cómo hacemos para que otras personas entiendan más fácil nuestros programas? ¿Podemos aprovechar soluciones creadas por otras personas para crear un nuevo programa?

En esta secuencia, a partir de un lenguaje simbólico con comandos para pintar casilleros en una grilla, las y los estudiantes se enfocan en las nociones de legibilidad y reutilización. También se recuperan las nociones de estrategia de solución, procedimientos y repetición simple.

Actividad 1

Las y los estudiantes escriben programas para hacer dibujos sencillos con el objetivo de motivar el uso de procedimientos y repetición simple.

Actividad 2

Las y los estudiantes escriben programas para un dibujo con partes diferenciadas, para recuperar la noción de estrategia y reflexionar acerca de la importancia de la legibilidad, es decir, de crear programas pensando en la facilidad de comunicación y comprensión del programa.

Actividad 3

Las y los estudiantes escriben programas para realizar una serie de dibujos similares entre sí, para introducir la noción de reutilización, asociada a la definición de procedimientos. Además, reutilizan los programas de sus compañeras y compañeros para poner de manifiesto cómo la reutilización y la legibilidad permiten la construcción colectiva de programas.

Actividad 4

A modo de actividad integradora, las y los estudiantes proponen un dibujo para que otro equipo escriba un programa que lo realice. Además de proveer una nueva instancia para aplicar las nociones abordadas, la elaboración de un desafío de programación lleva a reflexiones más profundas entre formas del dibujo y las herramientas de programación necesarias para conseguirlas. También, reforzamos la importancia de considerar cuestiones de legibilidad a la hora de compartir soluciones.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Soluciones a problemas computacionales

- Diseño de programas: estrategia, legibilidad.

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, secuencias, procedimientos, repeticiones.
- Ejecución secuencial de programas: ejecución, autómata.

Objetivos de aprendizaje

- Proponer soluciones que prioricen la legibilidad de los programas mediante la división en subproblemas y la definición y utilización de procedimientos y repeticiones.
- Incorporar la importancia de diseñar soluciones considerando la reutilización.

Saberes previos de CC

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, secuencias, procedimientos, repeticiones.
- Ejecución secuencial de programas: ejecución, autómata.

Eje: Soluciones a problemas computacionales

- Diseño de programas: estrategia, legibilidad.

Materiales necesarios

- Hojas de papel cuadriculado.
- Lápices o lapiceras.

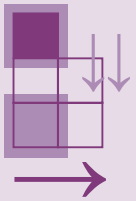
Actividad 1
Repeticiones y procedimientos

En esta actividad presentamos el lenguaje simbólico que las y los estudiantes utilizarán en la secuencia. Este lenguaje permite elaborar programas para dibujar en una superficie cuadriculada, y permite que hagamos énfasis en el uso de la repetición y la definición de procedimientos.

Objetivos >

Se espera que las y los estudiantes:

- Utilicen la repetición y los procedimientos para construir programas más fácilmente interpretables y modificables.



Inicio >

El **propósito de este momento** es introducir el lenguaje simbólico que utilizaremos en la secuencia y generar asociaciones con las nociones de autómata, primitiva y ejecución.

Orientaciones

Presentamos, a partir de un ejemplo, un lenguaje para dibujar pintando casilleros en una hoja cuadriculada, que emula los lenguajes de programación. Mostramos cuáles son las primitivas, cada una representada por un símbolo, y explicamos que permiten moverse un casillero para cada lado, o bien, pintar el casillero en el que se está situado.

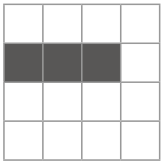
Primitivas	
←	Mover el lápiz al casillero de la izquierda
→	Mover el lápiz al casillero de la derecha
↑	Mover el lápiz al casillero de arriba
↓	Mover el lápiz al casillero de abajo
■	Pintar el casillero

Primitivas del lenguaje, con su símbolo y su significado.

Hacemos una breve secuencia de comprobación para esclarecer la dinámica de trabajo con estos programas.

Programa: ■ → ■ → ■

Resultado:



Un programa que pinta tres casilleros hacia la derecha, incluyendo el casillero inicial.

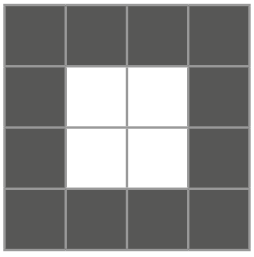
Es importante señalar a las y los estudiantes que, a pesar de que la ejecución la lleve a cabo una persona, dado su rol de autómata, solo podrá interpretar las instrucciones que estén formuladas según las reglas del lenguaje y ejecutar el programa tal cual está escrito (no podrá hacer cambios, aún cuando detecte que hay un error)[1].

Desarrollo >

El **propósito de este momento** es abordar desafíos que motiven el uso de repeticiones y procedimientos como estrategias de legibilidad.

Orientaciones

Dividimos la clase en grupos heterogéneos de dos o tres estudiantes.
El **primer desafío** consiste en escribir usando lápiz y papel un programa para realizar el siguiente dibujo (un cuadrado de cuatro casilleros de lado).
Mostramos a las y los estudiantes el resultado que se espera obtener y recordamos que tengan presentes las restricciones del lenguaje.



Luego de unos minutos de trabajo, les proponemos que compartan sus soluciones a los demás grupos.

■ → ■ → ■ → ■ ↓ ■ ↓ ■ ↓ ■ ← ■ ← ■ ← ■ ↑ ■ ↑ ■ ↑

Un programa posible para dibujar el cuadrado (iniciando en el primer casillero de la primera fila).

[1] Podemos retomar las nociones de instrucciones o comandos primitivos, ejecución y autómata abordadas en la secuencia "¿Qué es programar? Autómatas, programas y primitivas".

¿Hay alguna parte del programa que vean que se repite en sus soluciones? ¿Todos escribieron el programa de igual forma? ¿Qué semejanzas y diferencias hallan?

Con ese intercambio, entre todas y todos, se observará si se respetaron las restricciones y si identificaron o no patrones de repetición. Es posible que haya quienes hayan identificado los patrones sin necesidad de escribir toda la secuencia de instrucciones.

Por ejemplo, en la solución se repiten ■↓, ■←, y ■↑ tres veces consecutivas cada uno. A partir de esta observación, retomamos la noción de repetición y convenimos una forma de expresarla en el lenguaje que estamos usando. Por ejemplo, podemos acordar encerrar entre paréntesis los comandos que se quieren repetir e indicar cuántas veces debe ejecutarse a continuación. Por ejemplo, para expresar la secuencia ■→■→■→ que pinta tres casilleros a la derecha podemos escribir: (■→)3.

A continuación, las y los estudiantes reescriben o completan sus soluciones utilizando repetición.

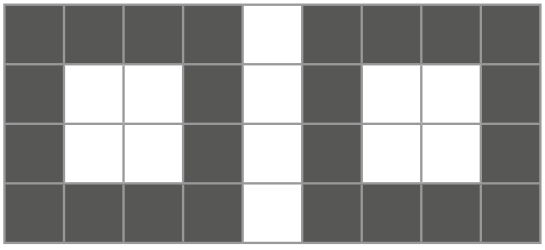
```
(■→)3
(■↓)3
(■←)3
(■↑)3
```

Un programa posible para dibujar el cuadrado escrito utilizando repeticiones.

Hacemos una puesta en común en la que les proponemos comparar las soluciones con y sin repetición.

Si no conocieran las dimensiones del cuadrado a dibujar, ¿en cuál de los programas lo podrían averiguar más fácilmente? ¿Por qué? Si tuvieran que cambiar el tamaño del cuadrado, ¿cuál de los programas sería más sencillo modificar?

Con este intercambio buscamos poner en evidencia el aporte de la repetición a la **claridad de un programa** y la **facilidad para modificarlo**.



El **segundo desafío** consiste en dibujar una figura compuesta por dos cuadrados de iguales dimensiones al del primer desafío. Mostramos a las y los estudiantes el resultado que queremos obtener.

El objetivo es recuperar el uso de procedimientos para aprovechar o reutilizar fragmentos de programas. Cuando aborden el desafío, es probable que surja rápidamente la necesidad de reutilizar un fragmento de programa existente. Cuando sea una preocupación compartida por todos los grupos, interrumpimos el trabajo para hacer una puesta en común y asociar esta necesidad con la definición de **procedimientos**.[2]

Como hicimos con la repetición, acordamos una manera de definir y utilizar procedimientos dentro de los términos del lenguaje que estamos usando. Es posible que algunos grupos ya hayan definido una forma propia; en ese caso, invitamos a que los grupos compartan sus propuestas para que sean consideradas.

Una forma de definir procedimientos puede ser escribir un nombre y seguido de dos puntos indicar los comandos del cuerpo de la definición. Una vez hecho esto, usaremos el nombre del procedimiento para invocarlo en el programa.

Una vez acordado cómo se expresarán los procedimientos, proponemos a los y las estudiantes reescribir el programa que dibuja los dos cuadrados.

```
Dibujar un cuadrado:
(■ →)3
(■ ↓)3
(■ ←)3
(■ ↑)3

Al ejecutar:
Dibujar un cuadrado
(→ ■)5
Dibujar un cuadrado
```

Una solución posible en la que se invoca dos veces al procedimiento "Dibujar un cuadrado" construido a partir del programa del desafío anterior.

```
Dibujar un cuadrado:
(■ →)3
(■ ↓)3
(■ ←)3
(■ ↑)3

Posicionarse en el segundo
cuadrado:
(→ ■)5

Al ejecutar:
Dibujar un cuadrado
Posicionarse en el segundo
cuadrado
Dibujar un cuadrado
```

Otra solución posible es tomar la repetición (→■) 5 y hacer un nuevo procedimiento que se llame "Posicionarse en el segundo cuadrado".

Al utilizar un procedimiento en distintos lugares del programa, es clave que las y los estudiantes tengan en cuenta en qué casillero de la grilla deberá estar ubicado el autómata para que el procedimiento consiga el dibujo esperado. Para ello, deberán incluir en el programa

[2] Para repasar esta noción, se puede consultar la secuencia "Definimos nuestros bloques. Procedimientos y repetición simple".

el desplazamiento necesario antes de ejecutar por segunda vez el procedimiento.

Comparen las soluciones que utilizan procedimientos con las que no y discutan: si no supieran cuántos cuadrados hay en la figura, ¿en cuál de los programas sería más fácil averiguarlo? Si tuvieran que cambiar esta cantidad, ¿cuál de los dos programas sería más fácil de modificar?

Al igual que con el desafío anterior, comparamos las soluciones para evidenciar y señalar cómo, al utilizar procedimientos, el programa se vuelve más fácil de interpretar y modificar.

Cierre >

El **propósito de este momento** es evidenciar que los programas requieren ser interpretados por las computadoras y las personas, y comenzar a construir la noción de **legibilidad**.

Orientaciones

¿Para qué partes de las soluciones de los desafíos les resultó conveniente utilizar repeticiones? ¿Por qué? ¿Y procedimientos?

Recuperamos el trabajo durante la actividad para identificar que, tanto al indicar repeticiones como al definir procedimientos para usar más de una vez, estamos evitando escribir muchas veces un mismo fragmento de programa.

¿Cuál es la ventaja de evitar fragmentos de programa repetidos?

Nos interesa destacar que estas herramientas, además de la ventaja evidente de ahorrarnos tiempo en escribir instrucciones, hacen que **los programas sean más fáciles de interpretar por las personas**. Esto resulta en que sean más fáciles de corregir (porque es más fácil leerlos para identificar errores), sean más fáciles de modificar (porque es más fácil comprender dónde y qué modificación hay que hacer) y también sean mejores para construir en equipo (porque es más fácil comprender los programas producidos por las otras personas para aumentarlos o mejorarlos).

¿Para quién escribimos programas: para las personas o para las computadoras? ¿Entonces cómo debemos escribirlos?

El dilema con el que concluimos este cierre busca resaltar que, si bien la computadora o el autómata tiene que ser capaz de interpretar el programa (y por eso es importante formular las instrucciones

con precisión y respetando los términos del lenguaje), también es fundamental que las personas puedan interpretarlos para estudiarlos, para continuarlos, para mejorarlos, para modificarlos, etc. Para ello, al programar, recurrimos a las herramientas del lenguaje que nos permiten escribir programas más claros o **legibles**. Entendemos por legibilidad a la cualidad de un programa de ser fácilmente interpretable por una persona.

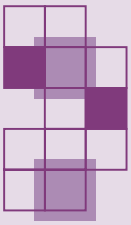
Actividad 2
Legibilidad

Continuando con la programación en papel y haciendo foco en la legibilidad de las soluciones, abordamos nuevos desafíos en los que la definición de procedimientos y su denominación será fundamental.

Objetivos >

Se espera que las y los estudiantes:

- Refuercen la legibilidad como una característica importante de los programas.
- Identifiquen el uso de procedimientos con denominaciones representativas y repeticiones para escribir programas que denoten una estrategia de solución.



Inicio >

El **propósito de este momento** es repasar brevemente las primitivas disponibles en el lenguaje con el que trabajamos en la **Actividad 1**, cómo se construyen los programas y las convenciones que definimos para expresar repeticiones y procedimientos.

Orientaciones

Invitamos a las y los estudiantes a recuperar estas ideas de su experiencia en la actividad anterior. Aprovechamos para conformar grupos heterogéneos pequeños.

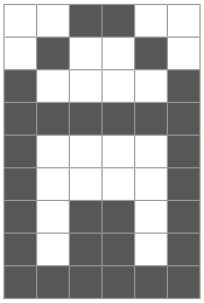
Desarrollo >

El **propósito de este momento** es reforzar la importancia de la legibilidad en la construcción de programas más extensos.

Orientaciones

Presentamos una nueva figura que representa una casa para escribir un programa que la dibuje.

Podemos comenzar por invitarlos a que nombren los tramos o partes a dibujar, ya que posiblemente los usen para nombrar los procedimientos, por ejemplo, “techo”, “pared”, “puerta”, “piso”.



Prestamos atención al trabajo de las y los estudiantes, observando (y alentando a) que **definan una estrategia de solución** y que **utilicen procedimientos** y repeticiones para expresar fielmente la estrategia.

A medida que van terminando, intercambian su programa con otro grupo para que lo ejecute para ponerlo a prueba. Quienes lo ejecuten deberán entregar el resultado.

```
Al ejecutar:
(↓)3
Dibujar el techo
Dibujar las paredes y el
piso
(↓)5
(→)2
Dibujar la puerta
```

```
Al ejecutar:
(↓)3
Dibujar pared izquierda
Dibujar piso y puerta
Dibujar pared derecha
Dibujar techo
```

Dos ejemplos (fragmentos) de soluciones posibles para dibujar la casa siguiendo distintas estrategias.

En el caso de que no se realice el dibujo esperado, podrá deberse a errores en el programa o a errores en la ejecución. En ambos casos, deberán registrar cómo pueden mejorar el programa. Por ejemplo, "El programa dibuja correctamente las paredes, pero el techo no está bien ubicado" o "El programa es correcto, pero para ejecutar sin equivocarse la parte que dibuja el techo hay que prestar mucha atención".

Cuando todas y todos hayan finalizado, invitamos a que compartan algunas de las soluciones que elaboraron para compararlas en una puesta en común. También, alentamos a que aquellos y aquellas estudiantes que hayan corregido errores en sus programas compartan cómo los identificaron y de qué manera los corrigieron.

¿En cuáles programas pueden anticipar el dibujo resultante? ¿Cuáles les parece que son más fáciles de ejecutar sin cometer un error? ¿Por qué? ¿En cuáles es más fácil saber en qué orden se dibuja cada parte de la casa? Si tuvieran que elegir uno para darle a alguien que no sabe qué figura hay que dibujar, ¿cuál elegirían?

Estas preguntas apuntan a analizar las soluciones compartidas para reforzar la importancia de la legibilidad. Señalamos que cuando podemos leer un programa y comprender sin mucho esfuerzo lo que hace, cómo está estructurado y ejecutarlo sin inconvenientes, podemos afirmar que el programa es **legible**. También señalamos que la legibilidad es importante para reflejar cómo quiso resolver el problema quien lo escribió, es decir, la **estrategia de solución** elegida. Esto se volverá

fundamental cuando trabajemos con programas escritos por otras personas (tanto para estudiarlos como para modificarlos).

Podemos brindar un tiempo para revisar y mejorar las soluciones incorporando estas ideas si hiciera falta.

Cierre >

El **propósito de este momento** es valorar la legibilidad de un programa para poder expresar fielmente la estrategia de solución elegida, facilitar la detección de errores e implementación de modificaciones, y permitir que otras personas puedan comprender con el menor esfuerzo posible su estructura y su propósito.

Orientaciones

Mostramos tres programas para interpretar a golpe de vista y anticipar cuál es el dibujo resultante.

Programa 1

```
Al ejecutar:
(■ ↑)7 ← ←(↓)3 ■ ↓ ■ → ■ ↓ ■ → → ↑
■ ↑ ■ → ■ ↑ ■ ↑ ↑ ← (■ ←)3 (■ ↑)3 (■ →)4 (↓ ■)2
(↑)3 (■ ← ←)2 ■
```

Programa 2

```
Al ejecutar:
Dibujar tallo
(←)2
(↓)3
Dibujar hoja izquierda
(→)2
↑
Dibujar hoja derecha
(↑)2
←
Dibujar pétalos

Dibujar tallo:
(■ ↑)7

Dibujar hoja izquierda:
■ ↓ ■ → ■ ↓ ■

Dibujar hoja derecha:
■ ↑ ■ → ■ ↑ ■

Dibujar pétalos:
(■ ←)3(■ ↑)3(■ →)4
(↓ ■)2
(↑)3 (■ ← ←)2 ■
```

Programa 3

```
Al ejecutar:
Dibujar una parte
(←)2
(↓)3
Dibujar parte izquierda
(→)2
↑
Dibujar parte derecha
(→)2
←
Dibujar el resto

Dibujar una parte:
(■ ↑)7

Dibujar parte izquierda:
■ ↓ ■ → ■ ↓ ■

Dibujar parte derecha:
■ ↑ ■ → ■ ↑ ■

Dibujar el resto:
(■ ←)3(■ ↑)3(■ →)4
(↓ ■)2
(↑)3 (■ ← ←)2 ■
```

Volvemos a mirar los programas para analizarlos en términos de legibilidad.

¿Pudieron averiguar qué dibuja cada programa y la manera en la que resuelve el problema? ¿Por qué?

Recuperamos la experiencia para identificar y valorar aquellas herramientas de programación que facilitaron la interpretación de los programas (procedimientos, repeticiones, división en subproblemas y la utilización de denominaciones representativas).

Para reforzar estas ideas podemos volver a mostrar los programas y observar que al ejecutarlos se llevan a cabo las mismas instrucciones en el mismo orden; la diferencia sustancial radica en la utilización y la denominación de los procedimientos para reflejar la **estrategia de solución** expresando la forma que eligió quien lo elaboró para resolver el problema.

¿Qué programa elegirían como definitivo para realizar este dibujo? ¿Por qué? ¿Y si tuvieran que compartirlo con otras personas? ¿Y si tuvieran que corregir algún error? ¿Y si tuvieran que modificar una parte?

El objetivo de estas preguntas es que, a modo de conclusión, las y los estudiantes recuperen las ventajas de los programas legibles que surgieron en reflexiones anteriores y las justifiquen para la elección del Programa 2. En este programa, por ejemplo, si se hubiera omitido un movimiento será más fácil encontrar el error porque ya sabemos qué debe hacer cada parte; o si quisiéramos modificar alguna de las partes del dibujo, por ejemplo, agregando pétalos o haciendo las hojas más grandes será más claro dónde hacer la modificación. En este sentido, valoramos la legibilidad porque al comprender lo que un código hace, significa que los programas sean mucho más fáciles de estudiar y **compartir** (son más fáciles de interpretar por otras personas), **corregir** (es más fácil encontrar dónde está el error) y **modificar o adaptar** (es más claro identificar qué sección del programa debemos cambiar).

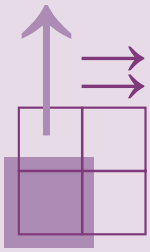
Actividad 3
Reutilización

Las y los estudiantes resuelven desafíos en los que se vuelve clave la definición de procedimientos para reutilizar fragmentos de programa.

Objetivos >

Se espera que las y los estudiantes:

- Reconozcan en problemas distintos un subproblema repetido y definan un procedimiento que lo resuelva para reutilizarlo en las soluciones a los otros problemas.
- Asocien la definición de procedimientos y su reutilización con la construcción colectiva de programas.



Inicio >

El **propósito de este momento** es recuperar, si hiciera falta, el lenguaje definido en las actividades anteriores y la modalidad de trabajo con los programas para dibujar, así como también la importancia de construir programas legibles.

Orientaciones

Invitamos a las y los estudiantes a que recuperen sus experiencias previas para recuperar estos acuerdos.

Desarrollo >

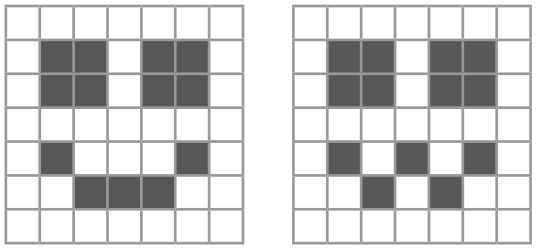
El **propósito de este momento** es resolver problemas reutilizando código propio y de otras personas, y reconocer la importancia de la definición de procedimientos para lograrlo.

Orientaciones

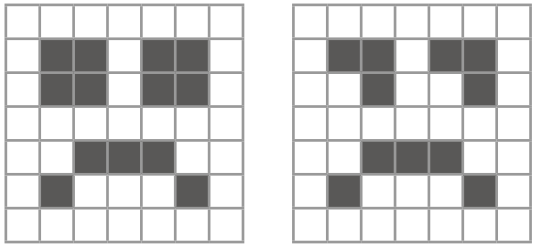
Proponemos la organización de la clase en pequeños grupos heterogéneos. Presentamos el nuevo desafío en el que cada grupo deberá dibujar dos "caritas". Podemos mostrar las opciones y que los grupos elijan, prestando atención a que estén todos los pares cubiertos. Aclaramos que cada grupo deberá hacer dos programas (uno para cada carita) y que pue-

den invocar en un programa cualquiera de los procedimientos que hayan definido en el otro.

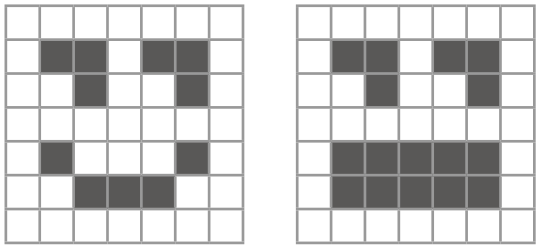
Par 1



Par 2



Par 3



Luego de un tiempo suficiente para la programación, se ponen en común los resultados y comienza el momento de reflexión.

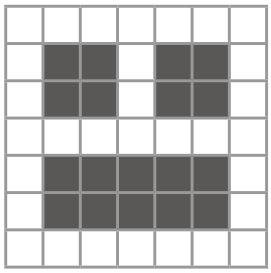
¿Definieron procedimientos? ¿Por qué? ¿Para resolver qué partes? ¿En qué otro/s desafíos de programación les pasó algo parecido? ¿Cuál es la diferencia con este problema?

Como primera aproximación a la idea de **reutilización**, identificamos que es conveniente definir procedimientos para las partes o subproblemas que se repiten. Podemos retomar experiencias previas con otros desafíos (por ejemplo, con el desafío ***Nuevos comandos*** de la secuencia "Definimos nuestros bloques. Procedimientos y repetición simple"), en las que definimos un procedimiento para un mismo subproblema se repitía dentro del desafío. En este deasfío, podría ser para dibujar cada uno de los ojos dos veces. Además, en esta actividad, aprovechamos **los procedimientos definidos para un problema** (la carita de la izquierda) **para resolver un problema nuevo** (la carita de la derecha).

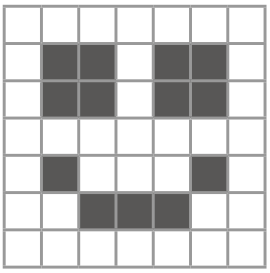
Si quisieran agregar un programa para que dibuje alguna de las caras que ya tienen, pero con un gorro, ¿qué podrían hacer para aprovechar al máximo lo que ya programaron?

Esta pregunta busca dar cuenta de que los **procedimientos son una herramienta fundamental para reutilizar** fragmentos de programas en programas distintos, que es la novedad que introduce esta actividad. En este caso, podría definirse un procedimiento que contenga el programa que ya hicieron para alguna de las caras y crear un programa nuevo que utilice ese procedimiento, además de uno nuevo "dibujar gorro".

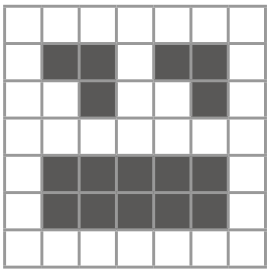
En una segunda instancia, con el propósito de generar **una situación en la que se ponga en juego la reutilización de programas como parte de una construcción colectiva**, proponemos a los grupos que piensen otros dibujos que podrían hacer en una cuadrícula similar, aprovechando los programas que fueron creados por todo el curso durante la actividad. Para esto, deberán relevar los programas elaborados por sus compañeros y compañeras y revisar la definición de los procedimientos en los propios para que otros grupos puedan reutilizarlos. Podemos registrar este relevamiento, por ejemplo, en el pizarrón, para que quede a disposición de todo el curso durante la actividad.



```
Al ejecutar:
Dibujar ojos cua-
drados
Ir al comienzo de
la boca
Dibujar boca
alargada
```



```
Al ejecutar:
Dibujar ojos
cuadrados
Ir al comienzo
de la boca
Dibujar boca
sonriente
```



```
Al ejecutar:
Dibujar ojos
felices
Ir al comienzo de
la boca
Dibujar boca
alargada
```

Ejemplo de cómo dibujar una nueva cara (izquierda) reutilizando procedimientos definidos en los programas preexistentes (derecha).

Luego, invitamos a que compartan los dibujos que imaginaron y los programas que los dibujan, señalando dónde intervienen los programas ya hechos, de quiénes los tomaron y cuál era su función original. Si se cuenta con el tiempo, proponemos que los grupos se intercambien los programas que escribieron, los ejecuten y luego corroboren el dibujo resultante. En caso de no obtener el dibujo esperado, alentamos a que los grupos conversen e identifiquen si se trató de un problema de ejecución o de programación e intenten corregirlo.

Cierre >

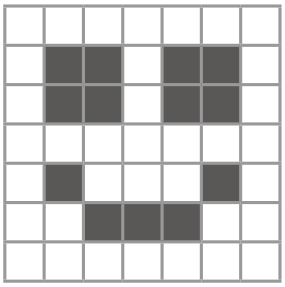
El **propósito de este momento** es recuperar la experiencia de la solución del desafío para asociar la legibilidad y la definición de procedimientos con la posibilidad de reutilizar programas hechos por otras personas para resolver nuevos problemas.

Orientaciones

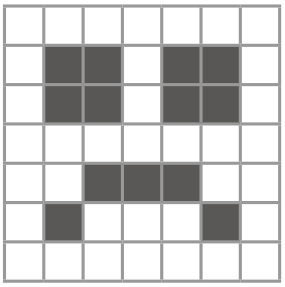
¿Qué características de los programas de las compañeras y los compañeros facilitaron la construcción de los nuevos programas?

A partir de esta pregunta, buscamos reforzar, por un lado, la importancia de la **definición de procedimientos** para aislar fragmentos de programas que pueden ser utilizados por otras personas y, por otro, que la **legibilidad** de un programa facilita su **reutilización** y, por lo tanto, la **construcción colectiva**.

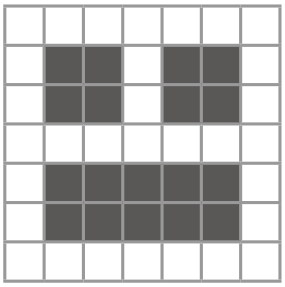
Con el propósito de evidenciar que como la **reutilización** favorece también la modificación y la mejora de los programas, mostramos los siguientes dibujos y los fragmentos de programas que los resuelven y pedimos que hipoteticen cómo habría que modificar los programas si quisiéramos que los dibujos tengan los ojos más grandes o de otra forma, y que comparen hacerlo en este caso, en el que hay un procedimiento que se reutiliza con el caso en el que cada programa tuviera las instrucciones completas para dibujar los ojos.



```
Al ejecutar:
Dibujar ojos cuadrados
Ir al comienzo de la boca
Dibujar boca sonriente
```



```
Al ejecutar:
Dibujar ojos cuadrados
Ir al comienzo de la boca
Dibujar boca triste
```



```
Al ejecutar:
Dibujar ojos cuadrados
Ir al comienzo de la boca
Dibujar boca alargada
```

En este caso, habría que modificar el procedimiento `Dibujar ojos cuadrados una única vez`, mientras que en el otro caso, habría que analizar tres programas y realizar tres modificaciones. La conclusión de esta reflexión es observar cómo la definición de procedimientos y

su reutilización hace que los programas también sean más fáciles de modificar: si el procedimiento necesita un cambio o una corrección basta con modificar el programa en un único lugar (en la definición del procedimiento). De la otra manera, para efectuar la modificación deberíamos inspeccionar la totalidad los tres programas, detectar cada vez que aparece el fragmento que queremos modificar y realizar las mismas modificaciones en todos estos fragmentos.

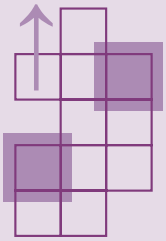
Actividad 4
Nuestros dibujos

Las y los estudiantes pondrán en práctica todos los conceptos tratados en las actividades anteriores a partir de pensar un dibujo para que otro grupo escriba un programa para realizarlo.

Objetivos >

Se espera que las y los estudiantes:

- Reconozcan qué características de un problema hacen que se vuelva clave la legibilidad y la reutilización en el programa que lo resuelve.
- Elaboren un programa legible para un problema aprovechando la reutilización de código.

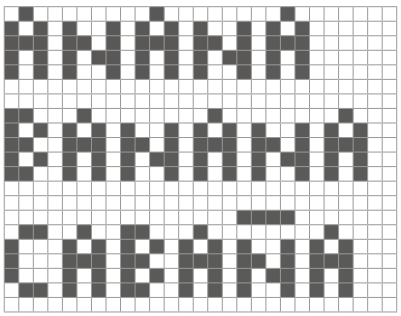


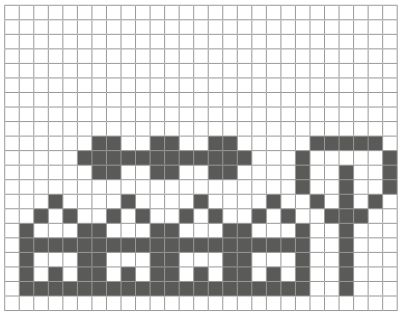
Inicio >

El **propósito de este momento** es identificar en la experiencia de las actividades anteriores qué características de los dibujos se asociaban con los conceptos trabajados en la secuencia (repeticiones, legibilidad y reutilización).

Orientaciones

En pequeños grupos heterogéneos, las y los estudiantes diseñan un dibujo que se pueda realizar en una hoja cuadriculada con el lenguaje utilizado en las actividades anteriores.





Ejemplos de dibujos que motivan la división en subproblemas y la reutilización.

Es importante señalar que al definir el dibujo tengan en mente una estrategia de solución para realizar el dibujo, anticipen la extensión del código

e incorporen patrones que habiliten repeticiones y reutilizaciones. Recorremos el trabajo de los grupos y sugerimos mejoras para reforzar estos aspectos.

Desarrollo >

El **propósito de este momento** es que las y los estudiantes pongan en práctica sus conocimientos de programación y, a la vez, profundicen las asociaciones entre características de los problemas y la necesidad de aplicar los conceptos abordados en la secuencia.

Orientaciones

En un **primer momento**, las y los estudiantes escriben un programa para realizar el dibujo que pensaron. En esta instancia es probable que detecten que el dibujo necesita modificaciones para que el programa incluya las características deseadas (estrategia en partes, patrones para reutilizar código, etc.). Es importante que, tanto el dibujo como el programa, no sean conocidos por el resto de los grupos y estén registrados en hojas o documentos separados.

En un **segundo momento**, cada grupo intercambia su dibujo con otro para que escriba un programa para realizarlo. A medida que las y los estudiantes escriben los programas, sugerimos observar e identificar el uso de procedimientos, nombres representativos y repetición. En el caso de observar la ausencia de algunos de estos conceptos, como motivación les podemos recordar que la solución será leída por otras personas para realizar el dibujo que pensaron. También es importante valorar aquellos programas donde estén presentes estos conceptos que aporten a la legibilidad.

En un **tercer momento**, a medida que los y las estudiantes finalizan sus programas, los deberán intercambiar con otro grupo para que lo ejecuten para obtener el dibujo resultante. Luego, deberán comparar el resultado obtenido con el dibujo esperado. Se alienta a que las y los estudiantes compartan su experiencia y valoren en conjunto aspectos positivos de los programas que favorecieron su comprensión (como la división en subtareas o la elección de nombres) y aquellas cosas que se podrían mejorar (y cómo las podrían mejorar).

En una segunda instancia, se comparan los programas que escribieron los grupos que recibieron el dibujo con los que elaboró el grupo que lo pensó para comparar estrategias. En este punto, es importante asociar características del dibujo con decisiones sobre los programas (por ejemplo, usar una repetición para un patrón repetitivo).

Esta actividad permite, por un lado, la percepción del avance en la programación en el grupo y en sus integrantes; pero además, facilita la inmersión en el concepto de legibilidad al comprobar si lo que un o una estudiante ha pensado en el desarrollo es interpretado de la misma manera por quien lo recepta o, al menos, hay una idea principal que queda clara en cada proyecto que se ha creado. Finalmente, la comparación entre los programas modelo y los obtenidos es una oportunidad para reforzar la asociación entre características de los problemas (en este caso, los dibujos) y los programas que los resuelven.

Retomaremos la estrategia de proponer la creación de problemas de programación para afianzar los conceptos ya abordados y profundizar las reflexiones sobre su uso asociado a problemas específicos en futuras secuencias, aprovechando el creador de desafíos de Pilas Bloques.

Cierre >

El **propósito de este momento** es brindar un espacio de metacognición para identificar en el recorrido de la actividad las instancias en las que tomaron decisiones que involucran los conceptos abordados en la secuencia.

Orientaciones

En un diálogo conjunto, rescatamos los distintos momentos de la actividad en los que se pusieron en juego las nociones abordadas en la secuencia. Invitamos a que relaten qué decisiones tomaron y asociadas a qué nociones.

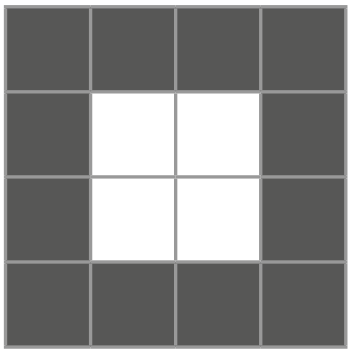
Esperamos que en el intercambio aparezcan las siguientes cuestiones.

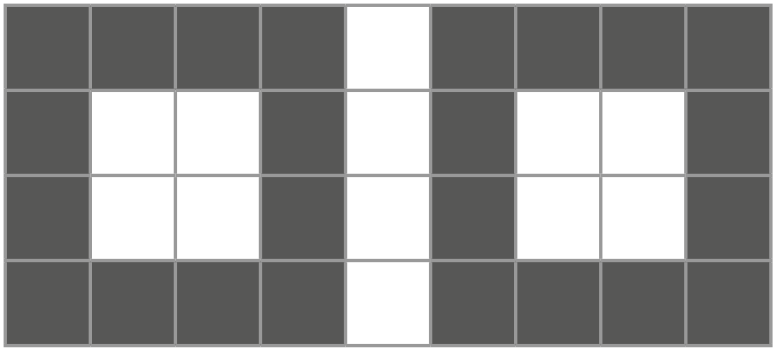
- Al diseñar el dibujo, incorporaron **patrones o elementos repetidos** que motivaron la reutilización de **procedimientos** y/o el uso de **repeticiones**. También, pueden haber elaborado un dibujo con distintos elementos o partes diferenciadas para motivar una **estrategia** con varias partes que refuerce la importancia de la definición de **procedimientos con nombres representativos**.
- Al recibir el dibujo tuvieron que identificar estas características en el problema y recuperar las nociones abordadas en la secuencia para escribir el programa.
- Al comparar el dibujo esperado con el resultante, es probable que hayan surgido errores que se debieron a que el programa no era tan fácil de interpretar. En la devolución al grupo que realizó el programa se pusieron en juego argumentos que involucran ideas de **legibilidad**.

- Al comparar el programa elaborado con el modelo, es probable que hayan surgido otras estrategias de solución más allá de la pensada por el equipo. Aprovechamos esta situación para poner de manifiesto las numerosas formas en las que se puede resolver un problema y cómo algunas de ellas aprovechan mejor determinadas herramientas y conceptos de programación.

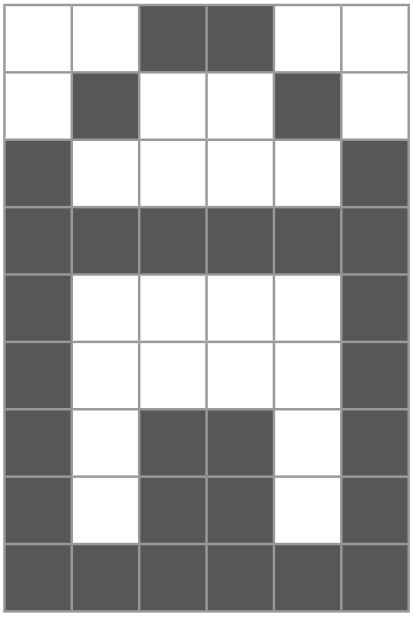
Anexo

Actividad 1





Actividad 2



Programa 1

Al ejecutar:
(■ ↑)7 ← ←(↓)3 ■ ↓ ■ → ■ ↓ ■ → → ↑
■ ↑ ■ → ■ ↑ ■ ↑ ↑ ← (■ ←)3 (■ ↑)3 (■ →)4 (↓ ■)2
(↑)3 (■ ← ←)2 ■

Programa 2

Al ejecutar:
Dibujar tallo
(←)2
(↓)3
Dibujar hoja izquierda
(→)2
↑
Dibujar hoja derecha
(↑)2
←
Dibujar pétalos

Dibujar tallo:
(■ ↑)7

Dibujar hoja izquierda:
■ ↓ ■ → ■ ↓ ■

Dibujar hoja derecha:
■ ↑ ■ → ■ ↑ ■

Dibujar pétalos:
(■ ←)3(■ ↑)3(■ →)4
(↓ ■)2
(↑)3 (■ ← ←)2 ■

Programa 3

Al ejecutar:
Dibujar una parte
(←)2
(↓)3
Dibujar parte izquierda
(→)2
↑
Dibujar parte derecha
(→)2
←
Dibujar el resto

Dibujar una parte:
(■ ↑)7

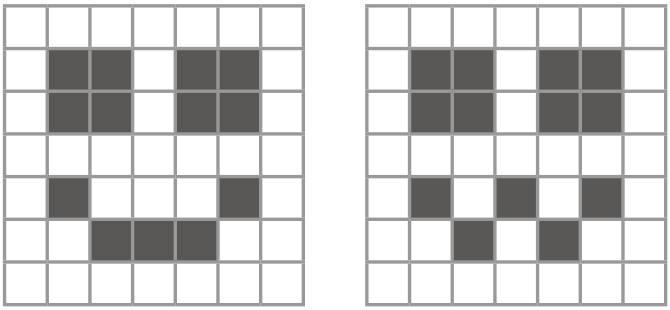
Dibujar parte izquierda:
■ ↓ ■ → ■ ↓ ■

Dibujar parte derecha:
■ ↑ ■ → ■ ↑ ■

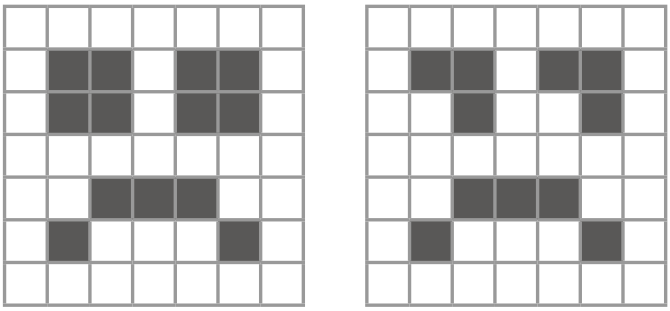
Dibujar el resto:
(■ ←)3(■ ↑)3(■ →)4
(↓ ■)2
(↑)3 (■ ← ←)2 ■

Actividad 3

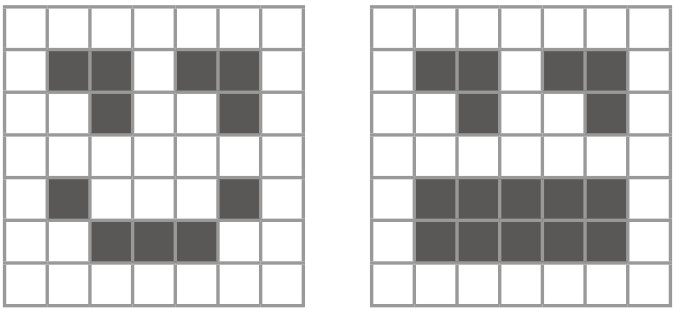
Par 1

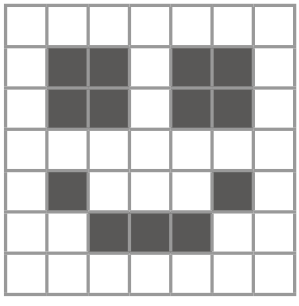


Par 2

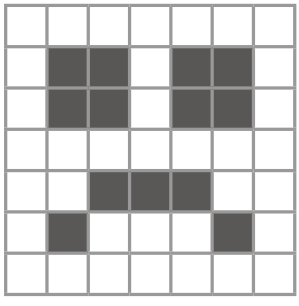


Par 3

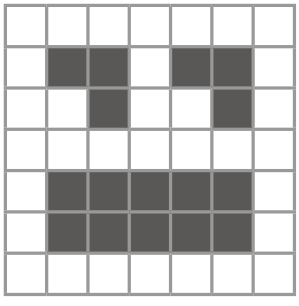




Al ejecutar:
Dibujar ojos cuadrados
Ir al comienzo de la boca
Dibujar boca sonriente



Al ejecutar:
Dibujar ojos cuadrados
Ir al comienzo de la boca
Dibujar boca triste



Al ejecutar:
Dibujar ojos cuadrados
Ir al comienzo de la boca
Dibujar boca alargada

Secuencia didáctica 4

Programamos estrategias en Pilas Bloques

Estrategia y división en subproblemas

¿Cómo abordamos desafíos más complejos? ¿Cómo hacemos para comunicar mejor estas soluciones?

En esta secuencia, las y los estudiantes resolverán desafíos más complejos en los que es clave la elaboración de una estrategia de solución por división en subproblemas y la definición de procedimientos con denominaciones representativas.

Actividad 1

A partir del desafío de Pilas Bloques ***Mañic en el cielo***, se recupera la noción de estrategia por división en subproblemas.

Actividad 2

A partir de los desafíos de Pilas Bloques ***Campeone desordenade*** e ***Yvoty y las luciérnagas***, se trabaja con la necesidad de construir procedimientos para realizar tareas que no están disponibles en las primitivas y la posibilidad de reutilizarlos.

Actividad 3

A partir del desafío de Pilas Bloques ***Reparadora de telescopios***, se aborda un desafío con primitivas limitadas, que obliga a repensar una estrategia de solución legible que contemple estas limitaciones, y un recorrido en dos dimensiones.

Actividad 4

El desafío de Pilas Bloques ***Mañic y los planetas*** motiva a las y los estudiantes a recuperar e integrar las ideas abordadas en las actividades anteriores de esta secuencia.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: procedimientos, repeticiones.

Eje: Soluciones a problemas computacionales

- Diseño de programas: estrategia, legibilidad.

Objetivos de aprendizaje

- Incorporar la elaboración de estrategias por división en subproblemas y su implementación en programas legibles que utilicen procedimientos.
- Reconocer patrones en el escenario que permitan reutilizar fragmentos de soluciones y aprovechar el bloque de repetición.
- Formular estrategias factibles en función de restricciones sobre las primitivas disponibles.
- Resolver problemas que involucran recorridos en dos dimensiones con programas legibles.

Saberes previos de CC

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, secuencias, procedimientos, repeticiones.

Eje: Soluciones a problemas computacionales

- Diseño de programas: estrategia, legibilidad.

Materiales necesarios

- Dispositivos con Pilas Bloques instalado o acceso a su versión en línea https://pilasbloques.program.ar/

Actividad 1
Mañic en el cielo

Se trabaja en esta actividad con un desafío de Pilas Bloques que tiene como propósito que las y los estudiantes reconozcan la necesidad de descomponerlo en subproblemas, utilicen el bloque de repetición simple y creen procedimientos.

Objetivos >

Se espera que las y los estudiantes elaboren una estrategia de solución a partir de la división de un problema en subproblemas más simples y la implementen en un programa legible.

Inicio >

El **propósito de este momento** es que las y los estudiantes exploren el desafío y comiencen a construir una solución que refleje sus aprendizajes previos.

Orientaciones

Organizamos la clase en grupos heterogéneos pequeños[1]. Invitamos a los y las estudiantes a ingresar al desafío ***Mañic en el cielo***. Les proponemos que lo resuelvan con las herramientas que han visto hasta el momento. Recorremos los grupos para identificar si requieren ayuda en el manejo de la herramienta y/o en la división de tareas para que todas y todos accedan a los dispositivos y puedan experimentar.

Escenario del desafío *Mañic en el cielo*.

[1] Se pueden encontrar propuestas de dinámicas lúdicas en el documento Enseñar computación desde la mirada de la Educación Sexual Integral (ESI).

Desarrollo >

El **propósito de este momento** es motivar a las y los estudiantes a identificar patrones para elaborar una estrategia por división en subproblemas y utilizar procedimientos y repeticiones para expresar esta estrategia en el programa.

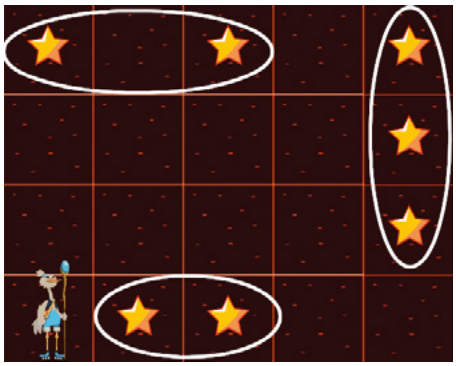
Orientaciones

Cuando las y los estudiantes hayan avanzado con la programación de la solución del desafío ***Mañic en el cielo***, podemos habilitar un intercambio oral en el que recuperemos de desafíos anteriores la importancia de la elaboración de una estrategia, acompañemos la identificación de subproblemas en el desafío que están trabajando, como así también enfaticemos la importancia de la legibilidad del programa para motivar la utilización de repeticiones y procedimientos.

> En estas primeras instancias de aprendizaje resultan especialmente valiosos los momentos de discusión, puesta en común y reflexión como una manera de ir construyendo los saberes esperados (en este caso, la división en subproblemas, la definición y la denominación de procedimientos y el uso de la repetición simple). Podemos proponer sucesivas pausas en el trabajo de programación para que los grupos compartan y analicen sus soluciones para identificar mejoras en estos aspectos.

Para recuperar la noción de división en subproblemas, en el intercambio oral, podemos promover el análisis del escenario para que reconozcan patrones en la distribución de las estrellas y, a partir de ello, logren dividir el problema en subproblemas. Podemos recuperar situaciones en las que han hecho esto, como en las actividades con el lenguaje para dibujar en papel cuadriculado de la secuencia "Programamos en papel cuadriculado. Legibilidad y reutilización" de esta colección o el desafío ***Chuy hace jueguito***, abordado en la secuencia "Definimos nuestros bloques. Procedimientos y repetición simple"[2].

[2] Las secuencias mencionadas son parte de la colección, que se encuentra disponible en el sitio curriculum.program.ar.



Una agrupación posible de las estrellas en el escenario que expresa la división del desafío en esos tres subproblemas.

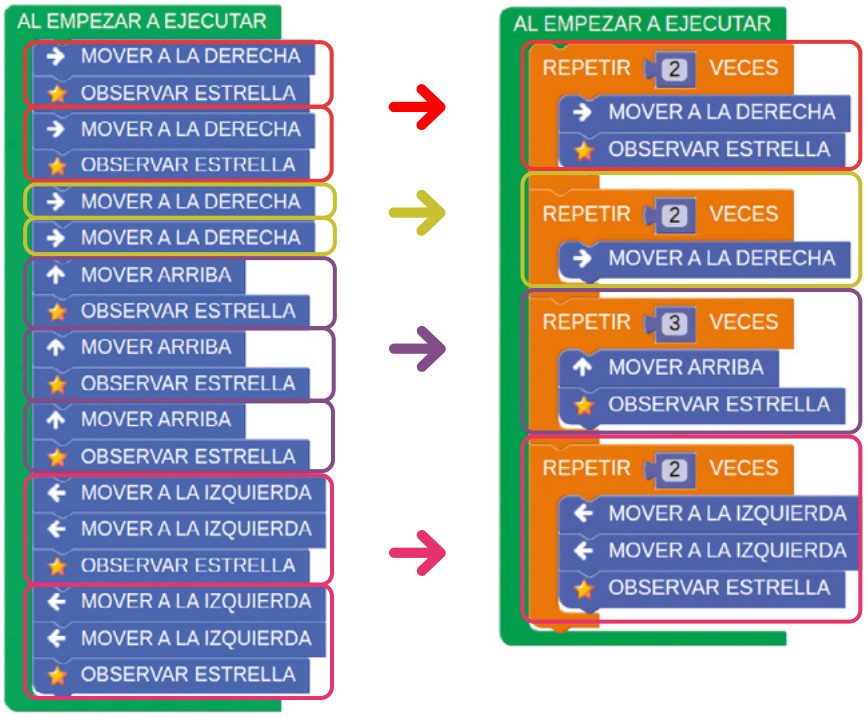
Para motivar el uso de repeticiones, es de gran ayuda identificar patrones de instrucciones que se repiten, teniendo presente la importancia de la legibilidad de la solución.

Fragmento de una solución que no aprovecha el uso de repeticiones ni procedimientos para mejorar la legibilidad.

Avanzamos hacia la importancia de definir procedimientos para reflejar la estrategia de solución y los subproblemas identificados. Es además una oportunidad para charlar sobre la conveniencia de dar un nombre a los procedimientos que describa el subproblema que resuelven. Para reforzar esta idea asociada a la legibilidad, podemos mostrar una solución sin procedimientos e hipotetizar la dificultad que tendría aumentarla o modificarla.

Si se trata de las primeras veces en las que las y los estudiantes interactúan con un desafío de estas características, tendremos que prestar especial atención y brindar una guía atenta para **evitar frustraciones tempranas**; también, enfatizar la recuperación de los conceptos de repetición y procedimiento para que tengan presente que esperamos que los incorporen a sus soluciones.

Para motivar el uso de la repetición simple, podemos ver en soluciones con y sin repeticiones qué patrones de instrucciones pueden reescribirse utilizando este bloque y cómo el programa resultante es más compacto y, por lo tanto, más legible, más fácil de interpretar.



Identificación de patrones en el programa que habilita el uso de bloques de repetición.

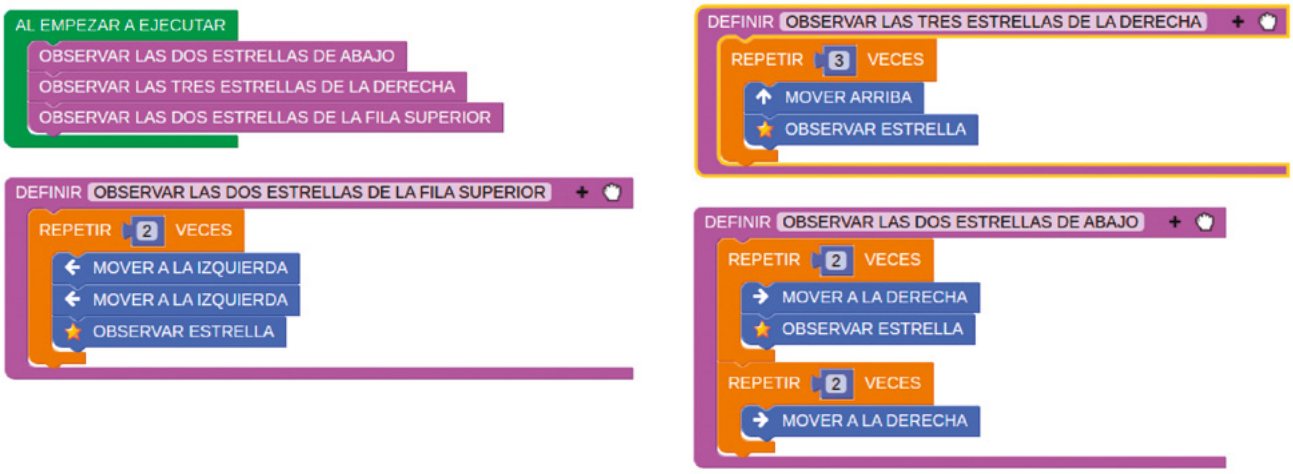
Cierre >

El **propósito de este momento** es compartir y comparar las soluciones para rescatar la existencia de múltiples estrategias posibles y, por lo tanto, la importancia de que esta estrategia esté claramente expresada en la estructura del programa (mediante el uso de procedimientos y repeticiones).

Orientaciones

Proponemos a los grupos socializar sus soluciones de manera que sus compañeros y compañeras puedan analizarlas. Orientaremos esta dinámica de manera de focalizarnos en:

- La existencia de múltiples estrategias posibles de solución (tanto en el modo de agrupamiento de las estrellas en el escenario como en el orden en el que se resuelve cada grupo de estrellas).
- La facilidad para reconocer, a partir de un programa, la estrategia pensada para resolver el problema y cómo la división en subproblemas facilita la comprensión de qué hace un programa.



Una solución posible, donde podemos observar la división en subproblemas (se recogen las estrellas en tres grupos, asociadas según su ubicación en los bordes del programa), el uso de procedimientos para reflejar esta estrategia en el programa y las repeticiones que aprovechan la forma de cada uno de estos grupos.

Podemos habilitar un espacio más o menos extenso para recorrer la clase mirando las soluciones de otros grupos y luego abrir la palabra para que los y las estudiantes cuenten qué han visto, cómo han agrupado las estrellas y en qué orden han abordado cada grupo. Luego, comentar cuáles soluciones les han parecido más legibles, cuáles les han resultado más claras o más complejas, etcétera.

El objetivo es arribar, entre todas y todos, a la conclusión de que identificar **subproblemas** y plantear una **estrategia**, definir **procedimientos** y utilizar **repeticiones** facilitan el proceso de elaboración de la solución y su comunicación con otras personas. Seguiremos aplicando y reforzando estas ideas en las actividades siguientes.

Actividad 2
Campeone desordenade + Yvoty y las luciérnagas

Con el desafío ***Campeone desordenade***, se motiva a las y los estudiantes a **recuperar la noción de estrategia**, la posibilidad de definir procedimientos y señalar los problemas de los **casos de borde en secuencias de repetición**. A continuación, las y los estudiantes resuelven de manera más autónoma el desafío ***Yvoty y las luciérnagas***, en el que vuelven a aparecer estas ideas.

Objetivos >

Se espera que las y los estudiantes:

- Reutilicen fragmentos de programa mediante el diseño de procedimientos que permitan varios usos o invocaciones.
- Definan procedimientos para generar bloques que resuelven tareas no disponibles como primitivas.
- Conozcan la noción de los casos de borde en las secuencias de repetición y su tratamiento.

Inicio >

El **propósito de este momento** es que las y los estudiantes exploren el desafío para identificar el problema del desplazamiento en diagonal y hacer un acercamiento sobre el caso de borde en una repetición.

Orientaciones

Invitamos a los grupos (conformados en la actividad anterior) a ingresar a Pilas Bloques para resolver el desafío ***Campeone desordenade***.

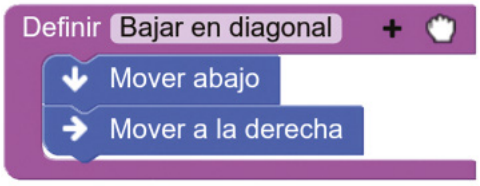
El escenario del desafío *Campeone desordenade* invita a identificar que los trofeos están dispuestos en una diagonal.

En una primera recorrida, vemos qué estrategias están desarrollando los grupos. Este desafío admite una solución muy sintética si se aprovecha que los trofeos están dispuestos en una diagonal; por ese motivo, es importante que las y los estudiantes lo identifiquen en su análisis del problema antes de continuar. Podemos motivar este enfoque si no surge espontáneamente invitándolos a identificar con preguntas (en una conversación con cada grupo o con toda la clase) que, a pesar de no estar dispuestos horizontal ni verticalmente, igualmente forman una hilera.

Identificado el patrón que estructurará la estrategia, es esperable que algunos grupos opten por pensar esa diagonal "hacia arriba" y otros "hacia abajo". En cualquier caso, se toparán con el problema de que no existe una primitiva que permita avanzar de un trofeo a otro moviendo a Chuy en diagonal. Dependiendo del nivel de avance que observemos, podemos conversar individual o grupalmente para arribar a la conclusión de que es necesario un procedimiento que realice este movimiento.

¿Cómo se mueve Chuy en la estrategia que pensaron? ¿Cuentan con una primitiva que realice este movimiento? ¿Recuerdan otros desafíos en los que no contaban con una primitiva para realizar un movimiento que debían hacer más de una vez?

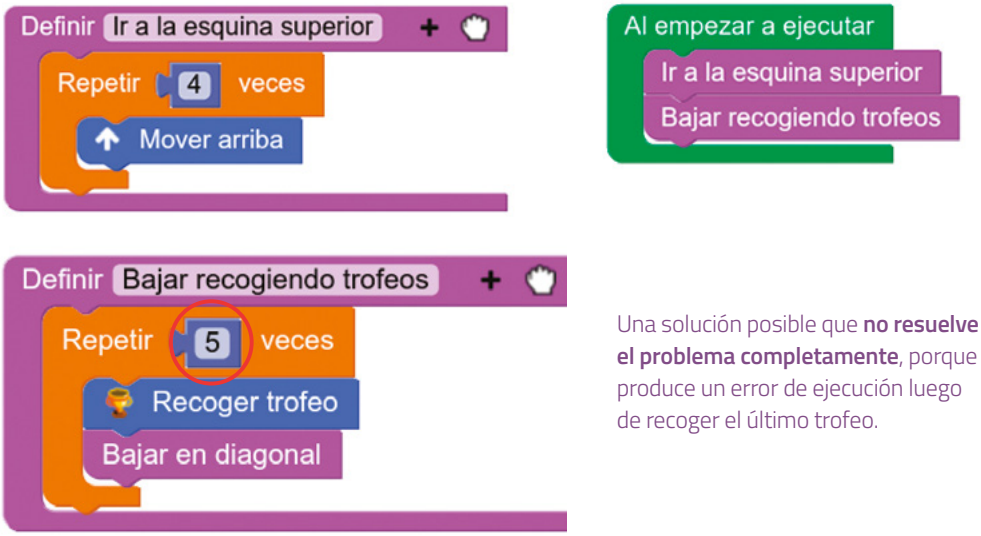
Apelamos a recuperar los desafíos anteriores (especialmente ***Nuevos comandos*** o ***Chuy hace jueguito***, trabajados en la secuencia "Definimos nuestros bloques. Procedimientos y repetición simple") en los que no existen primitivas para realizar movimientos o tareas que son centrales al problema e, incluso, que hay que realizar más de una vez. El objetivo es recordar que la manera de crear estos bloques es definiendo un procedimiento para que las y los estudiantes avancen en sus soluciones definiendo un procedimiento para mover al autómata en diagonal.



Un procedimiento posible para moverse en diagonal hacia abajo.

El próximo paso es continuar con la elaboración de la estrategia y la solución con este nuevo procedimiento creado. Una posible división en subproblemas es considerar, por un lado, desplazarse hasta el inicio de la diagonal de trofeos y, por otro, recogerlos. Ambos subproblemas apuntan a motivar a las y los estudiantes para que usen el bloque Repetir.

A medida que las y los estudiantes avancen, es probable que necesiten un nuevo refinamiento de la solución. Observando el tablero es probable que indiquen una repetición de 5 veces dado que hay 5 trofeos. Sin embargo, esta solución produce un error de ejecución en el último paso cuando se ejecuta el movimiento de Chuy hacia abajo.



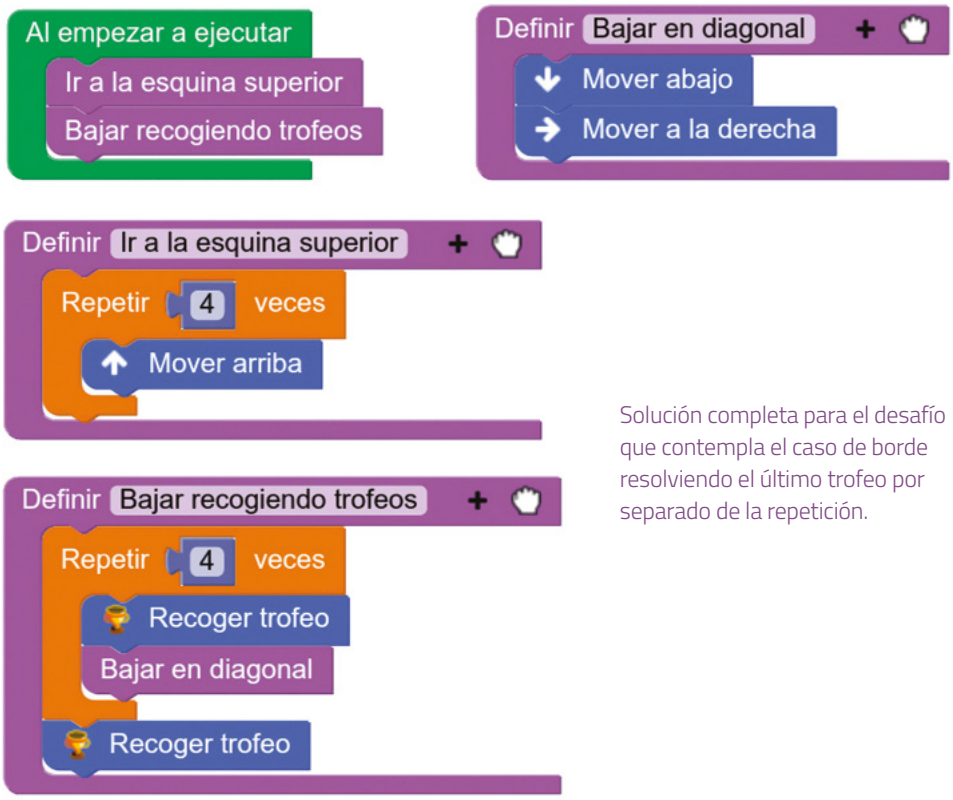
Una solución posible que **no resuelve el problema completamente**, porque produce un error de ejecución luego de recoger el último trofeo.

A medida que los grupos se encuentren con este inconveniente, invitamos a que corrijan sus programas. Para esto, podemos ayudarlos a observar o simular la ejecución paso a paso de la repetición para que identifiquen que el error sucede luego de recoger el último trofeo. Luego, abrimos un espacio de intercambio para que compartan el inconveniente encontrado y cómo lo resolvieron (o bien, en qué punto se trabaron y no pudieron resolverlo).

¿Cuándo ocurría el error de ejecución? ¿A qué se debía? ¿Cómo lo resolvieron? ¿Cuándo les parece que puede aparecer este tipo de inconvenientes?

En general, este tipo de inconvenientes sucede cuando debemos procesar un elemento y avanzar hacia el siguiente elemento, y esto se repite varias veces. El inconveniente se presenta al llegar al último elemento (o el primero, si comenzamos ubicados sobre él), ya que no se requiere que sigamos avanzando y, por lo tanto, no debemos repetir la acción completa de procesar y avanzar, sino solo procesar. Estos son **casos de borde de repetición**.

Conceptualizamos el inconveniente como un caso de borde de repetición: existe un patrón de elementos en el problema en el que el último elemento (o el primero) debe ser tratado de manera diferente (es decir, no es posible resolverlo con el contenido del bloque Repetir). Como una estrategia general, a partir de lo relatado por los grupos, podemos señalar que para resolver estos casos abordamos el último (o el primero) por fuera de la repetición y utilizamos el bloque Repetir para los demás elementos.



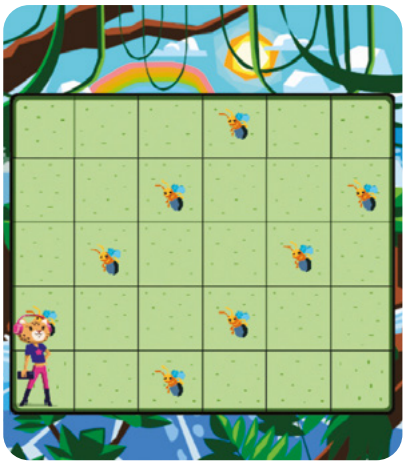
Solución completa para el desafío que contempla el caso de borde resolviendo el último trofeo por separado de la repetición.

Desarrollo >

El **propósito de este momento** es brindar una instancia de trabajo de mayor autonomía en la que las y los estudiantes pongan en juego las ideas abordadas en el inicio de la secuencia.

Orientaciones

Invitamos a las y los estudiantes a explorar el desafío ***Yvoty y las luciérnagas*** y comenzar a resolverlo.

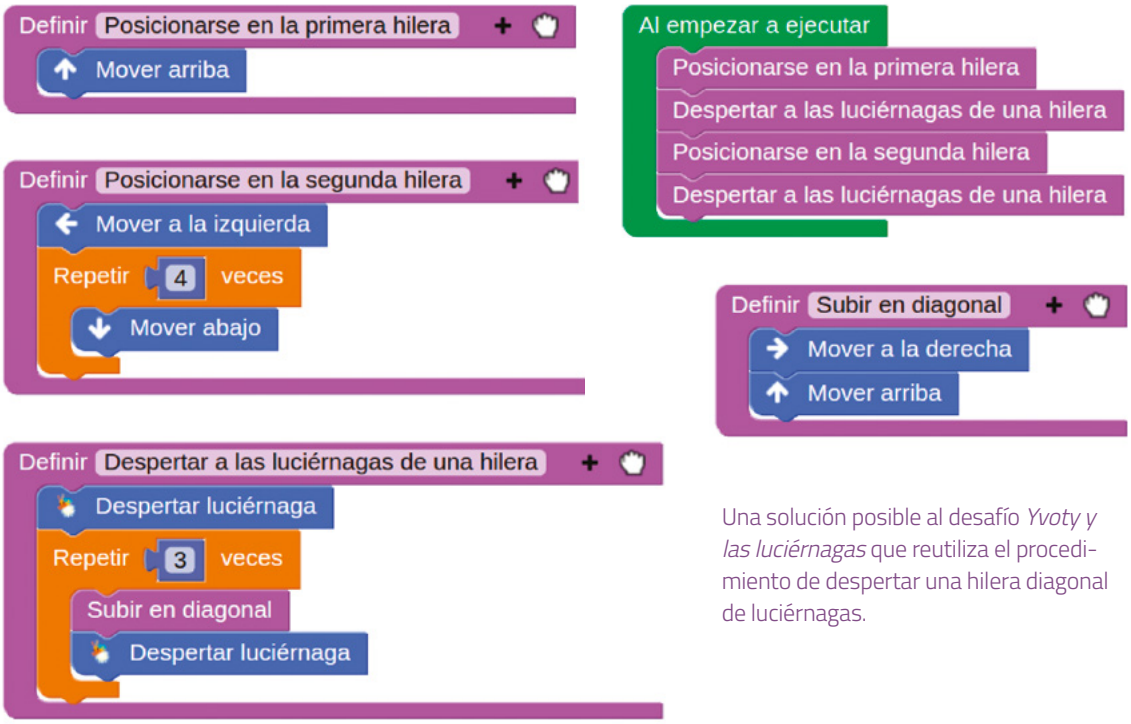


Escenario del desafío *Yvoty y las luciérnagas*.

Para quienes lo necesiten, podemos charlar sobre la disposición de las luciérnagas para que reconozcan que es similar a la de los trofeos del desafío anterior y, a partir de allí, motivar que recuperen las ideas abordadas en el inicio y transferirlas a este nuevo problema. En particular, apuntamos a que reconozcan la importancia del movimiento en diagonal y, por lo tanto, la necesidad de armar un procedimiento para agregar una funcionalidad que no estaba en la lista de primitivas.

Existen varias estrategias para resolver este problema, incluyendo algunas que no recorren las luciérnagas en diagonal (por ejemplo, despertar las dos luciérnagas de cada fila y pasar a la fila siguiente). Si lo consideramos pertinente, antes de que realicemos cualquier intervención, podemos invitar a los grupos a que compartan las estrategias que pensaron para compararlas y valorar las características que venimos trabajando (procedimientos definidos para agregar "funcionalidades" al desafío, una buena separación en subproblemas que sea fácil de comprender, la incorporación de repeticiones, la reutilización de partes del programa construido). Estas discusiones pueden ser insumo para una segunda instancia de trabajo en la que los grupos repiensen o mejoren sus soluciones.

Una solución particularmente interesante es aquella que aprovecha el hecho de que las dos hileras de luciérnagas son iguales y, por lo tanto, es posible resolverlas con un mismo procedimiento. Incluye la definición de un procedimiento para moverse un casillero en diagonal, subproblemas claramente distinguidos y fuertemente asociados a patrones en el escenario, nombres declarativos que explicitan la estrategia y el uso de repeticiones con casos de borde.



Una solución posible al desafío *Yvoty y las luciérnagas* que reutiliza el procedimiento de despertar una hilera diagonal de luciérnagas.

Si queremos hacer énfasis en la identificación de patrones y en la construcción de soluciones que aprovechen las regularidades del problema, podemos motivar la solución con preguntas que apunten a la identificación de las dos diagonales como subproblemas iguales y la definición de un procedimiento para resolverlos con la misma lógica de agregar al entorno bloques útiles para el problema que estamos resolviendo.

¿Cómo describirían en pocas palabras la disposición del escenario? ¿Cómo pensarían la solución si contaran con una primitiva llamada "Despertar hilera diagonal de luciérnagas"?

Cierre >

El **propósito de este momento** es explicitar en la experiencia de resolución de la actividad la noción de estrategia, la identificación de patrones, la definición de procedimientos y la posibilidad de reutilización de partes del programa para señalar sus ventajas en el proceso de programación.

Orientaciones

Invitamos a los y las estudiantes a compartir sus experiencias y sus soluciones para compararlas y debatir sobre ellas. Orientamos el intercambio poniendo la lupa sobre:

- La necesidad de **usar repeticiones** y la importancia de adquirir la práctica para identificar los patrones que nos permiten aprovechar

esta herramienta. Por ejemplo, las diagonales con los trofeos en la actividad anterior y las luciérnagas en esta actividad.

- La ventaja de **crear tareas que no están dentro de las primitivas** mediante el uso de procedimientos para facilitar el proceso de resolución. Por ejemplo, el procedimiento Subir en diagonal y la importancia de nombrar descriptivamente los procedimientos creados.
- La posibilidad de **reutilización** de partes del programa construido, mediante la definición de procedimientos usados más de una vez, por ejemplo, como se vio en el procedimiento para despertar una hilera de luciérnagas en el desafío ***Yvoty y las luciérnagas*** de esta secuencia o los procedimientos para realizar un mismo dibujo en la secuencia "Programamos en papel cuadriculado. Legibilidad y reutilización" de esta colección.
- La importancia de la **legibilidad** del programa construido para poder conocer cómo resuelve un problema, por ejemplo, en qué orden se reccorrían los grupos de estrellas en el desafío ***Mañic en el cielo***.

Actividad 3
Reparadora de telescopios

En esta actividad, las y los estudiantes ensayan una estrategia de solución para el desafío ***Reparadora de telescopios*** sin mirar las primitivas disponibles. Al observar las primitivas, probablemente, no encuentren algunas que habían utilizado y, por lo tanto, deberán adaptar la solución restringiéndose a las primitivas disponibles.

Objetivos >

Se espera que las y los estudiantes:

- Propongan una estrategia de solución restringida a las primitivas disponibles.
- Reconozcan que existe una relación entre las estrategias de solución posibles y las primitivas disponibles para implementarla.

Inicio >

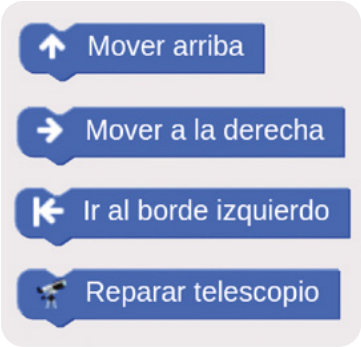
El **propósito de este momento** es que las y los estudiantes reconozcan que existe una relación entre las primitivas disponibles y las estrategias factibles para resolver un desafío.

Orientaciones

Mostramos el desafío ***Reparadora de telescopios*** e invitamos a los grupos a pensar una estrategia para resolverlo, con la salvedad de que *a priori* no revisen las primitivas ni los bloques disponibles.

Escenario del desafío *Reparadora de telescopios*.

Escuchamos diferentes estrategias y, a continuación, mostramos las primitivas disponibles. Probablemente, surjan situaciones de conflicto sobre cómo abordar la solución al ver que el autómata no posee los cuatro movimientos habituales: Mover arriba, Mover abajo, Mover a la derecha, Mover a la izquierda. A partir de este descubrimiento, pedimos a las y los estudiantes que revean sus estrategias de acuerdo con esta limitación.



Primitivas disponibles para resolver el desafío *Reparadora de telescopios*.

Desarrollo >

El **propósito de este momento** es resolver un desafío con primitivas restringidas.

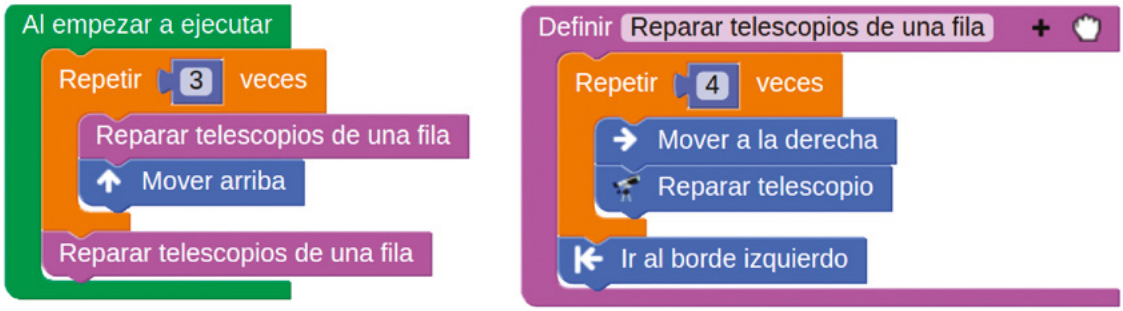
Orientaciones

Las y los estudiantes avanzan en la elaboración de la estrategia sujeta a las restricciones en las primitivas y la resolución del desafío.

Para quienes lo necesiten, podemos hacer algunas preguntas para su orientación.

¿Las primitivas disponibles nos "están diciendo algo" sobre la trayectoria que debe hacer Mañic? ¿Qué patrones ven en el escenario? ¿Qué subproblemas podemos identificar en el escenario?

Es importante que, además de la elaboración de la estrategia en este contexto particular, las y los estudiantes tengan presentes las ideas abordadas en los desafíos que trabajaron previamente en esta secuencia (principalmente, la definición y la denominación de procedimientos y el uso de repeticiones) y puedan aplicarlas con un mayor grado de autonomía en esta actividad. También podemos recordar el problema del caso de borde que vieron en la actividad anterior.



Una solución posible que consiste en repetir la resolución de una fila.

Cierre >

El **propósito de este momento** es reconocer que los lenguajes de programación (en este caso, a través de los bloques disponibles) imponen restricciones sobre las soluciones posibles; además, explicitar que este desafío implica un recorrido bidimensional y asociarlo con dos repeticiones.

Orientaciones

Invitamos a las y los estudiantes a compartir sus soluciones para comprobar que en este caso **la variedad de soluciones está limitada por las primitivas disponibles**.

¿Cuántas estrategias hay para un mismo problema? ¿De qué depende? ¿Hay algunas mejores que otras? ¿Puede haber algunas que no sean factibles?

Comparamos esta situación con desafíos anteriores en los que era posible elaborar estrategias diferentes y en las que cada grupo pudo elaborar una solución acorde a su idea o su estilo de solución. Podemos adelantar que esto ocurre habitualmente en la programación y una dificultad frecuente es **encontrar soluciones con las herramientas (limitadas) que nos brinda el entorno** o la herramienta que estamos usando para construirla.

Mirando el escenario, ¿pueden identificar una hilera de telescopios?, ¿cómo diría que están dispuestos? ¿Qué recorrido tiene que hacer Mañic para resolver el desafío? ¿Cómo se ve esto en la estrategia? ¿Y en el programa? ¿Cómo resultaría el programa si no utilizáramos procedimientos?

Otra conclusión que nos interesa señalar a partir de las soluciones planteadas es explicitar el hecho de que el problema consiste en realizar un **recorrido bidimensional**, ya que hay telescopios uno a continuación del

otro, tanto en dirección horizontal como vertical. Observando las estrategias podemos generalizar este tipo de recorridos como la repetición de **recorridos unidimensionales** (en este caso, repetimos la solución de una fila) que, a su vez, implica una repetición (para procesar cada uno de los elementos de la fila). Esto implica que el programa cuenta con **dos repeticiones** (una para hacer avanzar a Mañic por las filas y otra para hacer avanzar a Mañic dentro de cada fila), separadas en **procedimientos distintos**. De no definir el procedimiento, quedaría un bloque repetir dentro de otro, perjudicando la legibilidad.

Una solución que no define el procedimiento para resolver cada fila y, por lo tanto, es más difícil de interpretar.

Actividad 4
Mañic y los planetas

Se propone a las y los estudiantes recuperar, con un mayor nivel de autonomía, ideas de esta secuencia didáctica como estrategias de solución, procedimientos, repeticiones y problemas en dos dimensiones en la solución de un desafío de Pilas Bloques.

Objetivos >

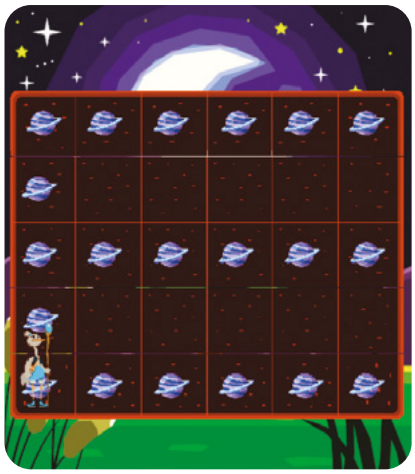
Se espera que las y los estudiantes propongan una solución que contemple la definición de procedimientos, separación en subproblemas, reutilización y estrategias de solución.

Inicio >

El **propósito de este momento** es presentar la situación problemática a resolver y que las y los estudiantes ensayen una estrategia de solución usando los conceptos vistos a lo largo de esta secuencia.

Orientaciones

Invitamos a los grupos a ingresar en el desafío de Pilas Bloques ***Mañic y los planetas*** para resolverlo. Como en los desafíos anteriores, alentamos una etapa de exploración y un ensayo para luego comenzar a construir una estrategia de solución.



Pantalla inicial del desafío *Reparadora de telescopios*.

Desarrollo >

El **propósito de este momento** es que las y los estudiantes resuelvan el desafío de forma autónoma.

Orientaciones

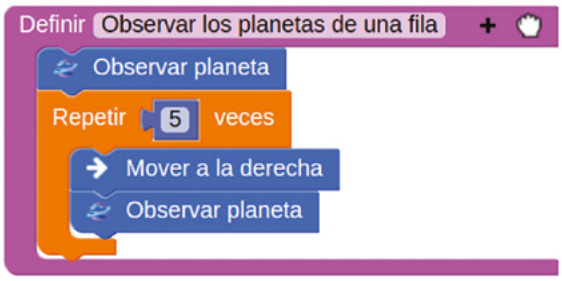
Mientras las y los estudiantes resuelven el desafío, recorremos los puestos de trabajo para alentar a que incorporen los conceptos principales trabajados en las actividades precedentes. Podemos brindar orientación con las siguientes preguntas.

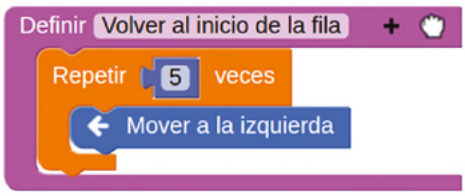
> *¿Qué figura dirían que forman los objetos en el escenario? ¿De qué manera podrían agruparlos? ¿Identifican algún patrón que se repita? ¿Habrá partes del programa que puedan reutilizar? ¿Probaron armar primero los procedimientos y luego ir completando de a uno?*

En el recorrido por los grupos, prestamos especial atención a:

- La estrategia de solución por **división en subproblemas** y su expresión mediante **procedimientos**.
- La identificación de **patrones** para dividir en subproblemas: identificar que los planetas forman un patrón de fila nos permite definir este subproblema en la solución
- La **reutilización** de procedimientos y uso de **repeticiones**.
- La construcción de **recorridos en dos dimensiones** con **repeticiones en dos niveles**: la repetición en el programa principal, que repite la solución para cada fila, y dentro de esta, las repeticiones de movimiento horizontal.

Una solución posible al desafío. Vemos la división en subproblemas y sus respectivos procedimientos en el programa principal, y también la identificación de patrones: las filas de planetas, subproblemas cuya solución se representa con procedimientos separados (uno para observar y otro para volver) que aprovechan el bloque de repetición. Identificamos el recorrido en dos dimensiones en las repeticiones de dos niveles: por un lado, la repetición en el programa principal, que repite la solución para cada fila, y, dentro de esta, las repeticiones que se mueve horizontalmente.





Cierre >

El **propósito de este momento** es habilitar un momento de metacognición para que las y los estudiantes identifiquen en su experiencia de programación los conceptos abordados.

Orientaciones

Invitamos a las y los estudiantes a socializar las soluciones del desafío que programaron conversando libremente en relación con el proceso de solución de los desafíos, las herramientas de programación usadas y las estrategias escogidas. Será particularmente interesante que puedan compartir ejemplos tomados de sus experiencias concretas durante la secuencia.

Nos interesa que logren explicitar:

- La importancia de pensar una estrategia para resolver el problema y, para ello, haber identificado los subproblemas, por ejemplo, cada fila de planetas en el desafío ***Mañic y los planetas*** o cada hilera diagonal de trofeos en el desafío ***Chuy Campeone desordenado.***
- La conveniencia de definir procedimientos para cada uno de estos subproblemas para explicitar la estrategia en el programa y hacerlo más legible para quienes lo construyen y para otras personas. Por ejemplo, en el desafío ***Mañic en el cielo***, poder reconocer con solo mirar el programa principal en qué orden el autómata observa los grupos de estrellas. Esta práctica también es relevante para ordenar la tarea de programación, ya que cuando completamos un procedimiento nos concentramos en la solución a un subproblema y podemos aislarlo (u olvidarnos) del problema completo que tiene mayor complejidad.
- La importancia de la definición de procedimientos para facilitar la interpretación de un programa para compartirlo con otras personas o encontrar errores si no funciona.

Secuencia didáctica 5

Creamos desafíos de repetición

Repetición simple

¿Cómo se usa el Creador de desafíos de Pilas Bloques? ¿Qué decisiones hay que tomar a la hora de crear un desafío? ¿Qué problemas se pueden resolver usando repetición simple?

Las secuencias didácticas basadas en los desafíos existentes de Pilas Bloques invitan a identificar un problema y proponer un programa que lo resuelva. El propósito de esta secuencia es invertir la propuesta: explicitada la característica que debe tener el programa, las y los estudiantes crearán un desafío en Pilas Bloques. Revisaremos para ello la noción de repetición simple con el objetivo de reforzar la relación entre las características de un problema y esta herramienta de programación necesaria para resolverlo.

Actividad 1

Las y los estudiantes se aproximan al Creador de desafíos de Pilas Bloques mediante la creación de desafíos sencillos para intercambiar con otros grupos que las y los invitan a explorar la herramienta.

Actividad 2

Las y los estudiantes crean desafíos que requieren el uso de repetición simple en la solución, para reforzar su conceptualización e identificar qué características de una situación problemática están asociadas al uso de esta herramienta.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Soluciones a problemas computacionales

- Diseño de programas: estrategias de solución.

Eje: Lenguajes de programación

- Herramientas de lenguaje de programación: repetición simple.

Objetivos de aprendizaje

- Familiarizarse con el proceso de creación de un desafío utilizando el Creador de desafíos de Pilas Bloques.
- Identificar, de manera más general o abstracta, relaciones entre la estructura de un problema y la necesidad de utilizar repetición simple en un programa que lo resuelva.

Saberes previos de CC

Área: Programación

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategias de solución.

Eje: Lenguajes de programación

- Herramientas de lenguaje de programación: repetición simple.

Materiales necesarios

- Dispositivos con Pilas Bloques instalado o acceso a su versión en línea https://pilasbloques.program.ar/

Actividad 1
¿Cómo se crea un desafío de Pilas Bloques?

Las y los estudiantes, trabajando en grupos, diseñan y construyen desafíos de Pilas Bloques que luego intercambian con otros grupos para su resolución.

Objetivos >

Se espera que las y los estudiantes se familiaricen con el proceso de creación de un desafío de Pilas Bloques utilizando el Creador de desafíos.

Inicio >

El **propósito de este momento** es poner de manifiesto qué es necesario definir para crear un desafío de Pilas Bloques.

Orientaciones

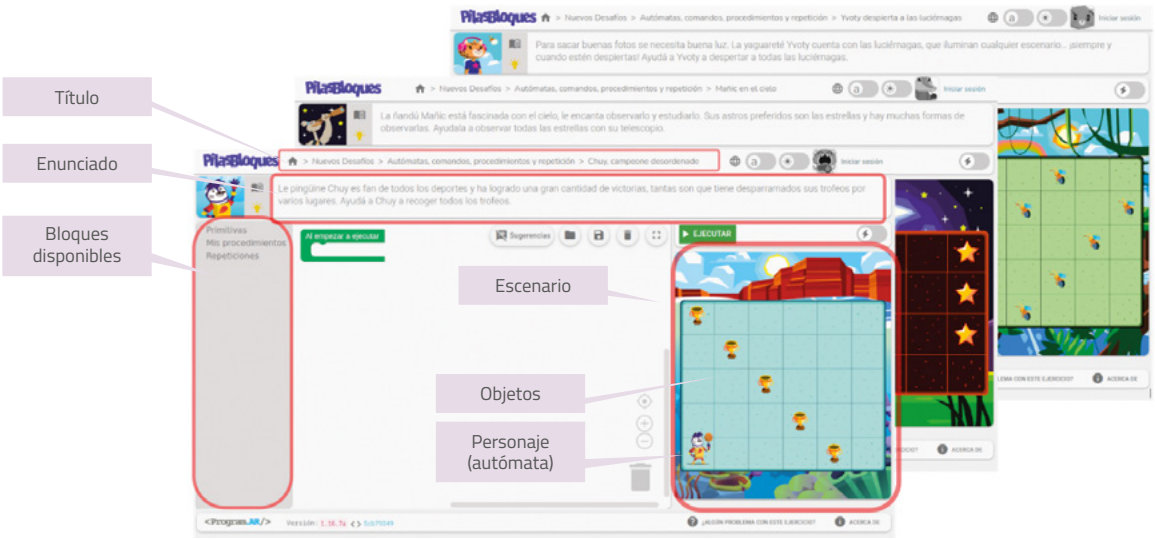
*Imagínense que son parte del equipo que crea desafíos en Pilas Bloques y les piden que **creen** un nuevo desafío. ¿Cómo es un desafío de Pilas Bloques?*

Enmarcamos toda la secuencia en un juego de rol en el que las y los estudiantes se desempeñan como miembros del equipo que desarrolla desafíos para Pilas Bloques. Los motivamos a que adopten un papel activo. Para identificar cómo están definidos los desafíos de Pilas Bloques, compararemos desafíos existen de diferentes personajes y con distintos bloques disponibles[1].

¿Qué diferencias pueden observar entre los desafíos? ¿Qué tenemos que lograr para resolver cada desafío? ¿De qué dependen los bloques disponibles en cada desafío?

[1] Por ejemplo, en Pilas Bloques, los desafíos "Campeone desordenade" y "Mañic en el cielo". Comparando estos desafíos se puede identificar qué elementos son comunes a todos los desafíos, y cuáles son propios de cada uno.

Invitando a participar a toda la clase para identificar en cada desafío: el personaje, el objetivo y cuáles son los elementos que siempre están y las variaciones en contenido o representación gráfica. La estructura común a todos los desafíos está compuesta por: un título, un enunciado (problema), los bloques disponibles, un escenario con obstáculos y objetos, y un personaje. Si fuera necesario, podemos reforzar la comparación eligiendo un nuevo desafío con otro personaje[2].



En esta secuencia no abordaremos herramientas de programación que requieran escenarios cambiantes. Por este motivo, no nos interesa explorar la posibilidad de definir más de un escenario y, por lo tanto, no lo consideramos entre los elementos a definir.

A partir de los elementos identificados en la comparación de los desafíos existentes, explicitamos en el pizarrón qué elementos hacen falta definir para crear un desafío:

- el personaje del desafío (entendiendo que cada personaje tiene sus propias acciones, que realiza con sus propios objetos disponibles);
- las dimensiones o tamaño del escenario;
- la ubicación de los objetos, los obstáculos y el personaje en el escenario;
- los bloques disponibles;
- el título del desafío, el enunciado y una pista opcional.

Es importante conservar estas anotaciones, servirán de guía para el resto de las actividades de esta secuencia.

[2] Por ejemplo, el desafío de Pilas Bloques "Yvoty despierta a las luciérnagas".

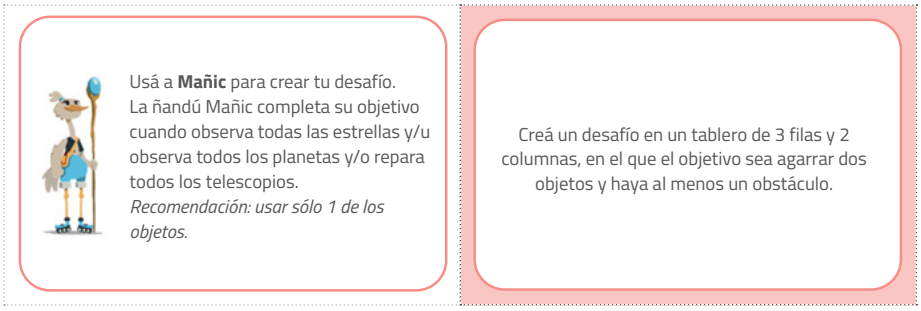
Desarrollo >

El **propósito de este momento** es que las y los estudiantes exploren activamente para **probar y descubrir** cómo funciona el Creador de desafíos de Pilas Bloques.

Orientaciones

Organizamos la clase en grupos heterogéneos de tres o cuatro estudiantes. Presentamos la consigna de la actividad a los grupos: cada uno creará un desafío de Pilas Bloques para que otro grupo lo resuelva.

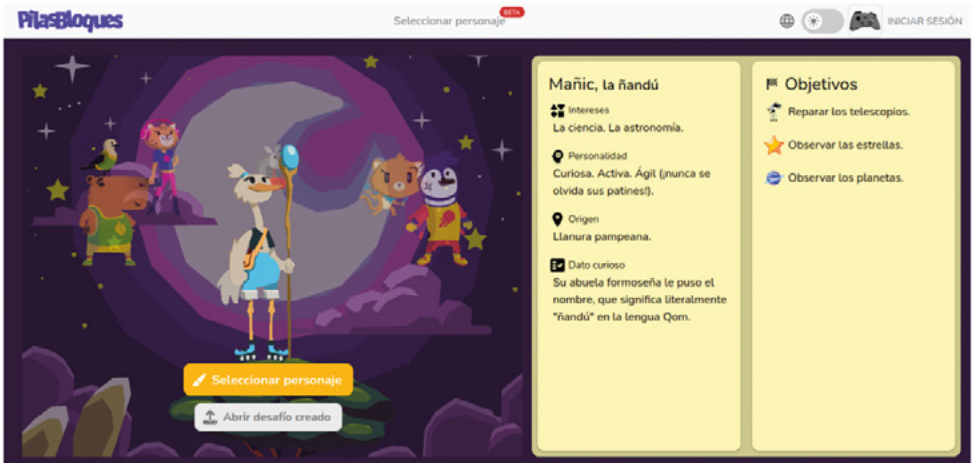
Para esto, se le entrega a cada grupo una tarjeta que define el personaje con el que deberán trabajar y su objetivo, y otra que describe requisitos del desafío a construir (ver Anexos I y II).



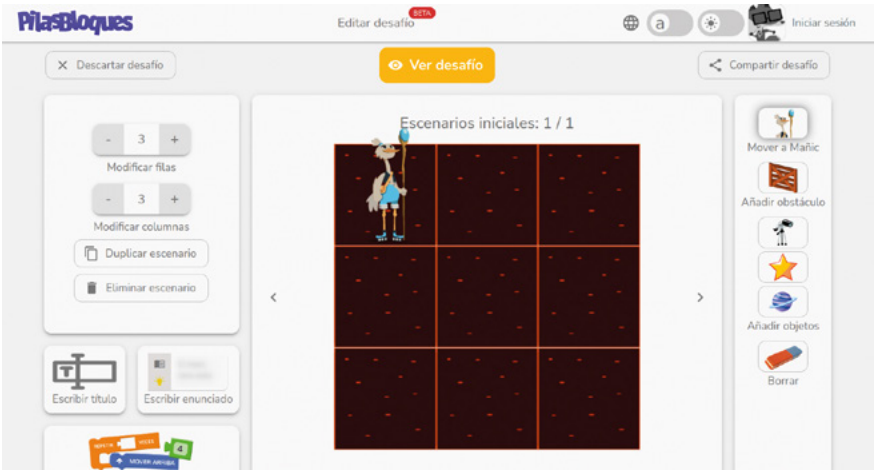
Ejemplo de par de tarjetas que podría recibir uno de los grupos.

Los invitamos a ingresar al Creador de desafíos de Pilas Bloques.

En la pantalla de inicio, deberán elegir la opción "Creá tu desafío".



En la siguiente pantalla, podrán elegir el personaje que les haya tocado en la tarjeta y ver más información sobre su historia y sus objetivos.



Luego de elegir el personaje, se accede al editor de desafíos.

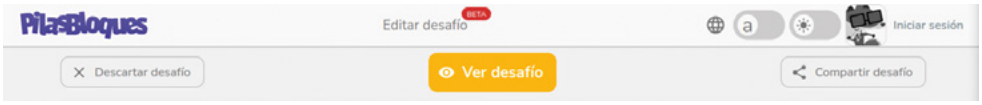
Para que las y los estudiantes puedan hacer el **recorrido completo de pensar, crear y probar un desafío** en un tiempo acotado, se pueden tener en cuenta las siguientes recomendaciones:

- Establecer un límite de tiempo motiva que no se detengan excesivamente en los aspectos visuales o elaborando desafíos innecesariamente complejos.
- A medida que evoluciona el trabajo de los grupos se puede reforzar que el desafío creado sea **lo más simple posible para cumplir con el objetivo de la tarjeta** (por ejemplo, se puede insistir en acotar la cantidad de objetos y obstáculos en el escenario y no agregar escenarios alternativos).

Mientras trabajan, es probable que requieran **asistencia en el uso de la herramienta**, sobre todo para agregar contenido al escenario o mover el personaje (que se realiza seleccionando en el panel derecho y luego haciendo clic en la ubicación).

A medida que los grupos vayan terminando, los invitamos a que **intercambien desafíos con otro grupo para su resolución**. Para esto pueden:

- elegir el modo "Ver desafío" y dar lugar a que otro grupo resuelva el desafío en la computadora.
- cliquear el botón "Compartir desafío" para exportar el desafío creado como un archivo y compartirlo para que otro grupo lo cargue en Pilas Bloques para resolverlo (en la ventana de inicio de Pilas Bloques al elegir "Importar desafío") o generar un enlace al desafío para que lo puedan compartir con el otro grupo. Ambas opciones pueden ser de utilidad si la clase se desarrolla en formato virtual sincrónico.



En la parte superior de la interfaz, aparecen los botones "Ver desafío" y "Compartir desafío".

En la pantalla de inicio, el botón "Importá tu desafío" permite cargar un archivo con un desafío para resolver.

Al terminar de resolver el desafío, el grupo que lo diseñó revisa si la solución era la que esperaba. En el caso de que no lo fuera, orientaremos a los grupos para que puedan darse retroalimentación sobre lo que se esperaba que sucediera y los inconvenientes que encontraron para lograrlo. Los alentamos a que juntos determinen los motivos por los que el resultado fue diferente a lo esperado. El grupo que creó el desafío realiza las correcciones necesarias. Luego, vuelven a ofrecer el desafío para su resolución.

Material para el docente

Algunas consideraciones sobre el intercambio de desafíos de Pilas Bloques

- Descargar los desafíos como archivos también es importante para que los grupos guarden su trabajo.
- Un desafío de Pilas Bloques posee la extensión de archivo ".dpbq" ("Desafío de Pilas Bloques"). Las soluciones tienen la extensión ".spbq" ("Solución de un desafío de Pilas Bloques").

Cierre >

El **propósito de este momento** es reforzar los elementos necesarios para definir un desafío de Pilas Bloques en el Creador de desafíos y reconocer el rol de las personas en el diseño y la creación de desafíos.

Orientaciones

Volvemos sobre la lista de elementos necesarios para definir un desafío de Pilas Bloques escrita en el pizarrón para que las y los estudiantes cuenten cómo hicieron para definirlos en el Creador de desafíos. Hacemos una puesta en común deteniéndonos en los siguientes aspectos.

***Sobre el escenario**. ¿Podían poner muchos obstáculos? ¿A algún grupo le pasó de "encerrar" su personaje o que hubiera un objeto inalcanzable? ¿Es necesario usar todos los objetos que tiene el personaje? ¿Tiene sentido construir un desafío sin objetos? ¿Y sin obstáculos?*

A partir de estas preguntas, esperamos que las y los estudiantes identifiquen la necesidad de proponer un desafío **que se pueda resolver** y que tenga **algún objeto** que funcione como objetivo, aún cuando sea simple, en el sentido de que el escenario cuente con pocos elementos.

***Sobre los bloques disponibles**. ¿Qué bloques decidieron poner? ¿Por qué? ¿Hay algunos que podrían no haber puesto? ¿Estaba mal poner todos? ¿En qué beneficia y en qué perjudica a quien resuelve el desafío? ¿Cómo se relacionan las soluciones posibles con los bloques disponibles? ¿Por qué querríamos elegir ciertos bloques y otros no?*

A partir de estas preguntas, nos interesa que las y los estudiantes identifiquen:

- **la necesidad de que ciertos bloques estén disponibles** y por qué son necesarios. Por ejemplo, si el personaje comienza en la derecha del escenario y su objetivo está a la izquierda, necesitará los bloques de movimiento a la izquierda, pero tal vez no los de movimiento a la derecha.
- Poner **muchos bloques disponibles** puede hacer más desafiante el ejercicio porque habrá que **elegir cuáles utilizar** en la solución y habilita **más formas de resolver el desafío**. Por ejemplo, si además de contar con los bloques de movimiento a la izquierda es posible moverse a la derecha, el personaje puede hacer más recorridos para llegar a su/s objetivo/s.
- Los **bloques disponibles determinan las soluciones posibles**. Por lo tanto, al elegir qué bloques ponemos como disponibles, estamos acotando o ampliando cuáles son las soluciones posibles. Al no ofrecer el bloque de movimientos a la derecha, estamos restringiendo las soluciones que involucran ida y vuelta y quien resuelva el desafío debe pensar una solución en la que el personaje solamente avance hacia la izquierda.

Para finalizar, identificamos que los desafíos de Pilas Bloques son creaciones que alguien pensó e hizo, tal como hicieron ellas y ellos, y que las decisiones que tomaron a la hora de crear sus desafíos son parecidas a las que tomaron las personas que crearon los desafíos de Pilas Bloques que resolvieron en clases anteriores.

Actividad 2
Nuestro desafío de Pilas Bloques para trabajar repetición simple

Proponemos a las y los estudiantes que piensen y creen un desafío cuyo objetivo sea abordar la repetición simple. Se replicará la dinámica de la **Actividad 1**, pero con objetivos que refieren a características de los programas que resuelven el desafío y no del desafío en sí mismo.

Objetivos >

Se espera que las y los estudiantes:

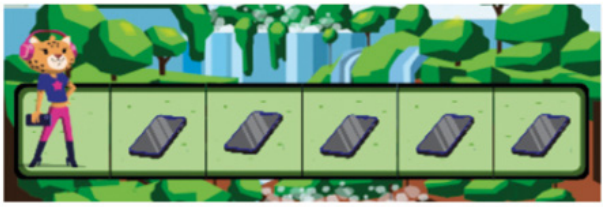
- Construyan una comprensión más profunda de la repetición simple a través del diseño de un desafío de Pilas Bloques que necesite de ese recurso.
- Identifiquen características generales de los problemas que requieren el uso de repeticiones simples.

Inicio >

El **propósito de este momento** es explicitar que existe una relación entre el diseño de un desafío y las nociones de programación que se propone trabajar.

Orientaciones

Evocamos la experiencia de la **Actividad 1** y mostramos el siguiente desafío, construido con el Creador de desafíos de Pilas Bloques, para que las y los estudiantes hipoteticen una solución posible.



¿Cuál dirían que es "el tema" que se aborda con este desafío? ¿Qué bloque es clave para resolverlo? ¿Qué decisiones de diseño del desafío hacen que esto sea así? ¿Qué cosas podrían cambiar y cuáles no para que siga siendo un desafío que requiere el bloque mencionado?

A partir de estas preguntas y el debate sobre las respuestas, interesa que los y las estudiantes identifiquen que:

- Un bloque clave para resolver este desafío es el bloque de repetición.
- Quienes diseñaron el desafío tomaron decisiones para que esto fuera así: decidieron que el problema del desafío requiriera repetir varias veces una misma acción (avanzar a la derecha y cargar el celular) y, por lo tanto, colocaron cinco celulares en el escenario.
- Podríamos cambiar el escenario del desafío y conservar esta intención si la solución sigue requiriendo la repetición de una acción una suficiente cantidad de veces como para que no sea factible hacerlo sin el bloque repetir (podrían ser 5, pero también 50 o 100).

Desarrollo >

El **propósito de este momento** es que las y los estudiantes reflexionen sobre qué características generales del escenario de un desafío de Pilas Bloques hacen que, al resolver el desafío, deban usarse repeticiones simples. Para ello, proponemos crear un desafío cuya solución requiera el uso de repetición simple (en vez de proponer uno dado para ser resuelto).

Material de referencia para docentes

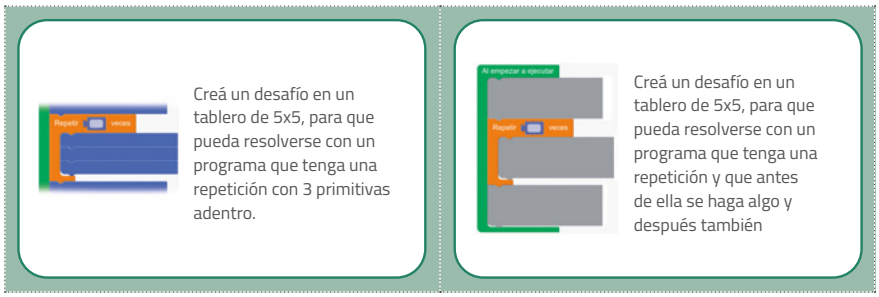
¿Podemos aprender a programar creando desafíos de programación?

Para resolver un desafío de Pilas Bloques, las y los estudiantes tienen que, a partir de un escenario y una consigna, construir un programa (esto involucra, en particular, definir una estrategia y utilizar determinadas herramientas de programación para implementarla). Al hacerlo, como docentes, nos interesa que asocien características de un problema de un nivel más general (por ejemplo, tiene partes claramente identificables o implica la repetición de una tarea puntual) con necesidades de un programa (contar con procedimientos definidos con nombres representativos o utilizar un bloque de repetición), con la intención de que puedan resolver situaciones problemáticas más generales que las de los desafíos puntuales que les presentamos.

En línea con el esfuerzo de reconocer generalidades en los problemas como una manera de avanzar en la construcción de los programas que los resuelven, proponemos invertir el camino: a partir de una característica de un programa, las y los estudiantes deberán proponer un desafío que se resuelva con un programa con esta característica. Si antes nos podíamos preguntar "¿**Necesitaremos repetición** para resolver este problema?", ahora debemos respondernos "¿**Cómo debe ser un problema** para que su solución involucre el uso de repetición simple?". La respuesta a ambas preguntas implica una comprensión profunda tanto de la herramienta de programación como de su aplicación en la solución de situaciones problemáticas.

Orientaciones

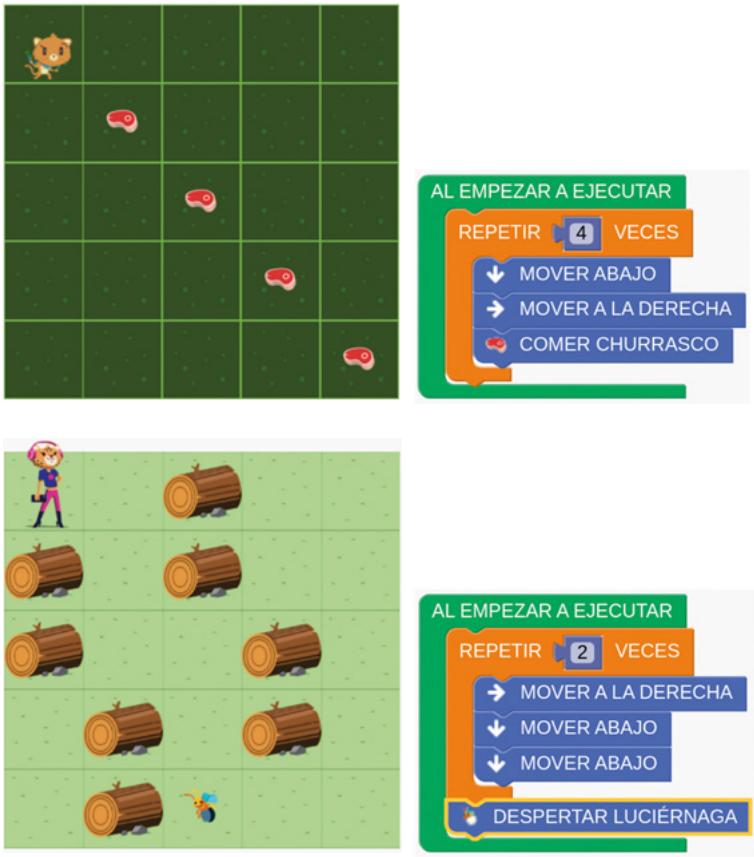
Organizamos la clase en grupos heterogéneos pequeños. Les entregamos una tarjeta (Ver Anexo III) para que creen un desafío con un personaje de su elección. Aclaramos que la dinámica de la actividad consistirá en crear un desafío que sea resuelto por otro grupo.



Ejemplos de tarjetas a entregar a los grupos.

Tenemos que tener presente que tarjetas de esta actividad buscan motivar en las y los estudiantes reflexiones que profundicen la comprensión del uso de la repetición simple para resolver desafíos.

- La **tarjeta que requiere tres comandos dentro de la repetición** apunta a que las y los estudiantes reconozcan el uso de la repetición para resolver desafíos en el que un subproblema aparece varias veces consecutivas y reconozcan la forma de un patrón en el escenario que pueda ser resuelto con tres primitivas.



Dos ejemplos de escenarios que cumplen con el objetivo de esta tarjeta con patrones que se pueden conseguir con tres primitivas. En el primer caso, la repetición incluye dos primitivas de movimiento (a la derecha y abajo) y una de interactuar con un objeto; en el segundo, incluye tres primitivas de movimiento (a la derecha, abajo y abajo).

- La tarjeta que **requiere acciones antes y después de la repetición** apunta a que las y los estudiantes identifiquen que el desafío debe contar con tres partes y qué parte del escenario corresponde resolver utilizando repetición. Además, cómo esto debe reflejarse en qué bloques que se colocan antes, dentro y luego del bloque de repetición en el programa.

Dos ejemplos de desafíos que cumplen con esta tarjeta. En el primer caso, el primer paso es recoger una lata moviéndose a la derecha. Esta primera parte aparece en el programa como las tres primitivas que están antes del bloque de repetición. Luego, la repetición incluye las primitivas para mover a los personajes al borde superior y, finalmente, se completa el desafío con la acción de levantar el papel a la derecha de los protagonistas mediante las primitivas que aparecen después del bloque repetir. Algo análogo ocurre en el segundo caso, en el que antes del bloque repetir aparecen las primitivas para reparar el telescopio y dirigirse hacia la fila con las estrellas, luego la repetición con las primitivas para observar las tres estrellas alineadas avanzando hacia la derecha y finalmente el traslado y la tarea de observar un planeta.

Teniendo en cuenta estas ideas para poder orientar a los grupos, los invitamos a que comiencen a pensar el desafío. Recorremos los grupos prestando atención a las inquietudes y los avances. Es probable que necesitemos acompañar el trabajo de cada grupo individualmente, especialmente para ayudarlos a identificar **cuáles son las características clave que deben incorporar al desafío** para cumplir con el objetivo pedido.

- Es importante acompañar a cada grupo con preguntas orientadoras para que **reflexionen sobre cómo deben ser los escenarios** de los desafíos para cumplir con los objetivos de la tarjeta.
- A aquellos grupos que no sepan cómo comenzar a abordar el desafío, les podemos sugerir **refinar el boceto de la forma del programa** que aparece en la tarjeta para avanzar hacia una solución abierta, pero más concreta sobre la que seguir trabajando luego en Pilas Bloques.
- Para mayor orientación, les podemos proponer que consulten **desafíos que ya hayan resuelto** con programas similares usando repeticiones.

El próximo paso es **intercambiar los desafíos para que sean resueltos** por un grupo diferente al que los creó y que haya trabajado con un objetivo diferente (es decir, con una tarjeta diferente)[3]. Esto se puede **realizar cuando consideremos que el desafío está listo, pero también se puede alentar a hacerlo previamente como una iteración que sea parte del proceso de creación** si se detectan dificultades en el diseño mien-

[3] Ver opciones para gestionar el intercambio de desafíos y soluciones en la **Actividad 1**.

tras se acompaña el trabajo de los grupos. Por eso, es importante que los grupos arriben rápido a una primera versión del desafío que se pueda probar (tanto por ellas y ellos mismos como por otro grupo) para lo cual no deberán diseñar escenarios muy complejos.

A medida que los grupos van resolviendo los desafíos creados, alentamos intercambios entre los grupos para motivar reflexiones que permitan asociar la disposición del escenario y los bloques disponibles del desafío con las soluciones posibles, en línea con los propósitos de cada tarjeta. Podemos hacer preguntas que los invite a hipotetizar variaciones del desafío con la intención de explicitar las decisiones que tomaron para que el resultado cumpla el objetivo, tanto para que identifiquen dificultades o errores como para que expliciten sus razonamientos en el proceso de elaboración.

Cierre >

El **propósito de este momento** es brindar una instancia de metacognición para identificar que en el proceso de construcción de los desafíos fue necesario descubrir relaciones entre la disposición del escenario y los bloques disponibles con las características del programa que lo resuelve. También, es explicitar estas relaciones y generalizarlas, teniendo en cuenta la importancia del análisis del problema a la hora de proponer un programa para solucionarlo.

Orientaciones

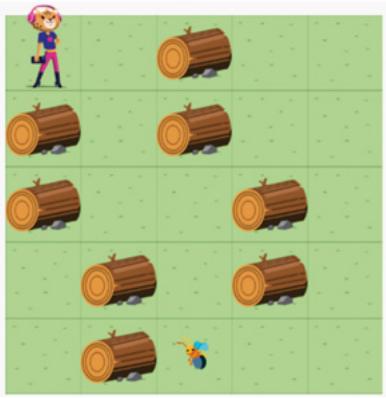
Invitamos a los grupos a que compartan sus desafíos y que relaten las decisiones que tomaron y el proceso iterativo que realizaron con el grupo que lo probaba. De esta experiencia nos interesa señalar que **la complejidad de la actividad residía en encontrar una relación entre la forma del desafío (el escenario y los bloques disponibles) y las soluciones posibles** y rescatar cómo la fue construyendo cada grupo. Podemos agregar a esas discusiones que este ejercicio es similar al que hacen cuando resuelven un desafío ya creado (es decir, buscar relaciones entre el problema que tienen que resolver y las herramientas de programación que van a utilizar para resolverlo), pero que hacerlo "al revés" permitió concentrarse más profundamente en este aspecto. Por lo tanto, forma parte de una instancia más profunda de su aprendizaje de programación.

Para explicitar las relaciones encontradas a partir de cada tarjeta e intentar expresarlas de manera más general, podemos partir de la **comparación de desafíos que cumplan con el objetivo con otros que no**.

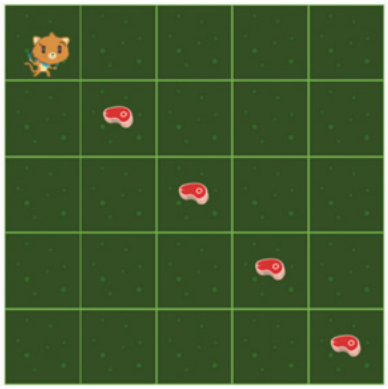
Podemos considerar tanto las versiones finales de los desafíos como las intermedias (lo que nos permite recuperar los motivos por los cuales los desafíos tuvieron que ser mejorados) o desafíos de ejemplo que no formen parte de la producción de los grupos. **El objetivo de esta instancia es construir un espacio de debate y análisis, atendiendo a que ningún grupo se sienta expuesto o atacado ni se menosprecie el trabajo de nadie.**

Para la tarjeta que requiere una repetición con tres primitivas

Podemos citar los ejemplos que mostramos en el desarrollo y recuperar los elaborados en la actividad para, por un lado, explicitar que la repetición está asociada con un mismo subproblema que aparece repetido en el escenario y, por otro, señalar cómo se relacionan las primitivas dentro del bloque Repetir con el patrón que resuelven en el escenario. En particular, con tres primitivas de las disponibles para estos personajes, podemos construir tres acciones de movimiento que resultan en recorridos con patrones de tres casilleros (ejemplo A) o dos acciones de movimiento y una de interactuar con un objeto que resultan en patrones de objetos a dos movimientos de distancia (ejemplo B).

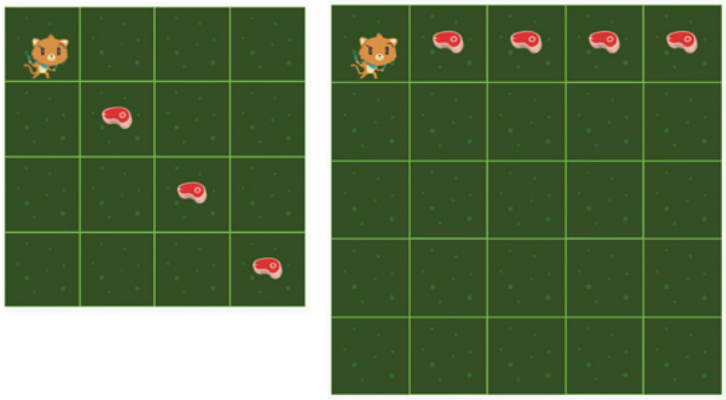


Ejemplo A.



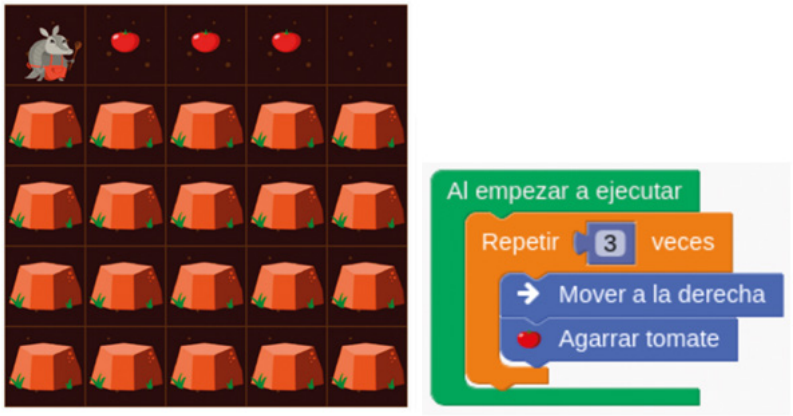
Ejemplo B.

Para reforzar estas ideas podemos hipotetizar modificaciones del escenario que harían que se siga cumpliendo el objetivo y otras que no. Por ejemplo, para una de las tarjetas: cómo modificarían el escenario para que el programa siga requiriendo un bloque de repetición simple con tres primitivas, pero con distinta cantidad de repeticiones o qué modificarían del escenario si el bloque de repetición debiera tener dos primitivas en vez de tres.



Variaciones para el desafío de Duba con los churrascos. A la izquierda, se mantiene el objetivo de las tres primitivas dentro de la repetición, pero cambia la cantidad de repeticiones. A la derecha, la repetición incluye únicamente dos primitivas (avanzar y comer).

Otra manera de reforzar estas ideas es confrontar estos escenarios exitosos con otros que no cumplan los requisitos de la consigna.



Un desafío (y una posible solución) que no cumple con el objetivo de la tarjeta de requerir una repetición con tres primitivas.

Para la tarjeta que requiere una repetición con primitivas antes y después

Como ejemplos exitosos podemos recuperar los citados en el desarrollo para generalizar la idea de una estrategia con tres partes, qué primitivas corresponden a cada una y, por lo tanto, cuáles deben estar abarcadas dentro del bloque de repetición.

Siguiendo la misma lógica del caso anterior, podemos plantear escenarios en los que no se cumple la consigna como una manera de reforzar las generalizaciones.

Un escenario y una solución posible que revela que no cumple la consigna ya que no hay ninguna acción "antes" de la tarea repetitiva.

También se puede jugar con la variación, proponiendo qué se debería cambiar en un escenario para que el desafío necesite dos instancias de repetición, intercaladas con dos de "hacer otra cosa".

Un escenario y una solución posible a la nueva consigna que requiere dos repeticiones intercaladas. Las acciones son reparar el telescopio (no requiere repetición), observar las estrellas horizontales (requiere repetición), observar el planeta (no requiere repetición) y bajar y observar las estrellas verticales (requiere repetición). Vemos cómo los comandos correspondientes a estas acciones figuran intercalados en el programa[4] y la importancia de cuáles están alcanzados por los bloques de repetición y cuáles no.

[4] Presentamos una solución sin procedimientos para poder observar todas las primitivas en secuencia. Sin embargo, en una solución modelo esperaríamos nombrar estas cuatro partes.

Anexo I
Tarjetas de personaje

Usá a **Mañic** para crear tu desafío.
La ñandú Mañic completa su objetivo cuando observa todas las estrellas y/u observa todos los planetas y/o repara todos los telescopios.
Recomendación: usar sólo 1 de los objetos.

Usá a **Chuy** para crear tu desafío.
Le pingüine Chuy completa su objetivo cuando recoge todos los trofeos y/o patea todas las pelotas y/o usa todas las paletas.
Recomendación: usar sólo 1 de los objetos.

Usá a **Capy y Guyrá** para crear tu desafío.
El carpincho Capy el picabuey Guyrá completan su objetivo cuando recogen todas las latas y/o recogen todos los papeles.
Recomendación: usar sólo 1 de los objetos.

Usá a **Yvoty** para crear tu desafío.
La yaguareté Yvoty completa su objetivo cuando despierta todas las luciérnagas y/u observa todas las mariposas y/o desbloquea todos los celulares y/o carga todos los celulares.
Recomendación: usar sólo 1 de los objetos.

Usá a **Duba** para crear tu desafío.
La puma Duba completa su objetivo cuando se come todos los churrascos.

Usá a **Lita** para crear tu desafío.
La mulita Lita completa su objetivo cuando recoge todas las lechugas y/o recoge todos los tomates y/o prepara todas las ensaladas.
Recomendación: usar sólo 1 de los objetos.

Anexo II
Tarjetas de desafíos de la Actividad 1

Creá un desafío en un tablero de 3 filas y 2 columnas, en el que el objetivo sea agarrar dos objetos y haya al menos un obstáculo.

Creá un desafío en un tablero de 2 filas y 3 columnas en el que el personaje empiece del lado derecho de la pantalla y deba llegar a un objeto esquivando obstáculos.

Anexo III
Tarjetas de desafíos de la Actividad 2

Secuencia didáctica 6

Seguimos programando estrategias en Pilas Bloques

Estrategia y restricciones de orden

¿Podemos ejecutar cualquier primitiva en cualquier momento? ¿Cómo influyen en la estrategia de solución del problema las restricciones sobre el orden en el que deben ejecutarse las primitivas?

En esta secuencia se profundiza la noción de **estrategia** a partir de la solución de desafíos que involucran acciones que deben realizarse en un orden determinado. Interpretamos estas restricciones bajo la noción de **estado**.

Actividad 1

A partir de los desafíos de Pilas Bloques ***Cargando los celus*** e ***Instalando juegos***, las y los estudiantes identifican la noción de estado y cómo condiciona la elaboración de estrategias al imponer un orden a ciertas acciones.

Actividad 2

A partir del desafío de Pilas Bloques ***El gran escape en yacaré***, las y los estudiantes analizan la importancia de utilizar procedimientos para proponer estrategias de solución a problemas con varias etapas.

Actividad 3

Para resolver el desafío de Pilas Bloques ***Limpiando el humedal***, las y los estudiantes integran el uso de procedimientos y repeticiones, la elaboración de estrategias de solución por división en subproblemas y su implementación en programas legibles.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Soluciones a problemas computacionales

- Diseño de programas: estrategia, legibilidad.

Eje: Soluciones a problemas computacionales

- Herramientas de lenguajes de programación: comandos primitivos, secuencias, procedimientos, repeticiones.

Objetivos de aprendizaje

- Proponer e implementar estrategias de solución que combinan comandos primitivos, procedimientos y repeticiones, para problemas de varias etapas con restricciones en el orden de las acciones, priorizando la legibilidad del programa.

Saberes previos de CC

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, secuencias, procedimientos, repeticiones.
- Soluciones a problemas computacionales.
- Diseño de soluciones computacionales: estrategia, legibilidad.

Materiales necesarios

- Dispositivos con Pilas Bloques instalado o acceso a su versión en línea https://pilasbloques.program.ar/

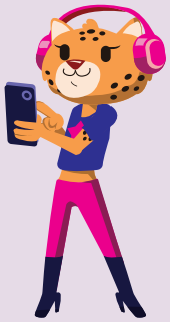
Actividad 1
Cargando los celus + Instalando juegos

Introducimos a las y los estudiantes la noción de estado al ver que en algunos desafíos se requiere que algunas acciones se realicen en un orden determinado.

Objetivos >

Se espera que las y los estudiantes:

- Reconozcan restricciones en el orden en el que algunas primitivas deben ejecutarse y propongan estrategias de solución que lo consideren.
- Se aproximen a la noción de estado como la caracterización de un momento de la ejecución del programa en función de qué acciones es posible realizar a continuación.



Inicio >

El **propósito de este momento** es explicitar que existen restricciones sobre el orden en el que deben realizarse algunas acciones.

Orientaciones

Dividimos la clase en grupos heterogéneos pequeños. Una vez formados los grupos, las y los estudiantes ingresan al desafío de Pilas Bloques ***Cargando los celus*** e intenta resolverlo sin darles indicaciones adicionales. El objetivo de la exploración es que observen que no es posible ejecutar la primitiva para cargar un celular si antes no fue recogido el cargador.

Error de ejecución al tratar de ejecutar la primitiva Cargar celular si antes no se cumplió con la tarea de recoger el cargador.

En una breve puesta en común, proponemos que las y los estudiantes recuperen los errores de ejecución que observaron para poner de mani-

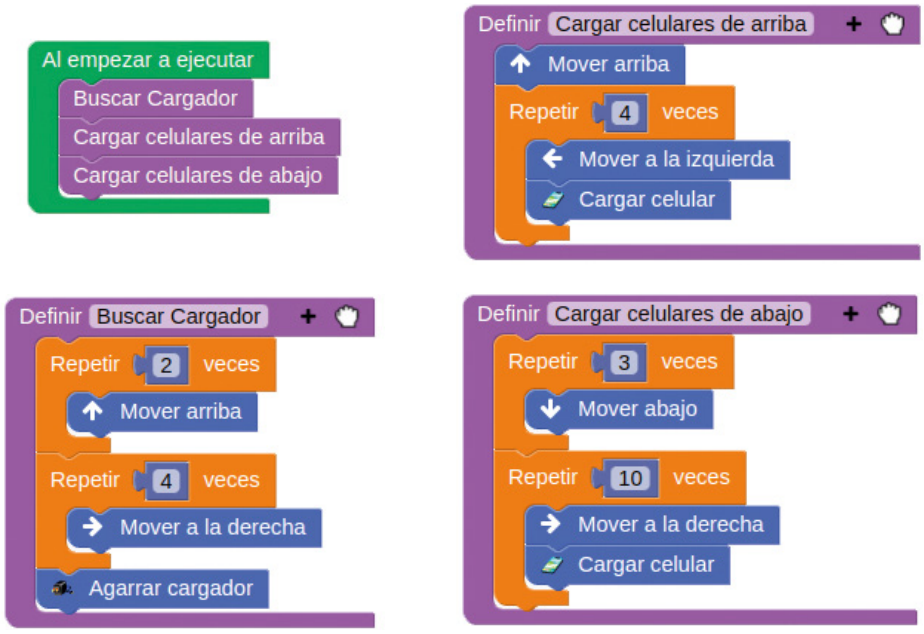
fiesto que este desafío requiere que se realicen ciertas acciones antes que otras.

Desarrollo >

El **propósito de este momento** es presentar un problema cuya solución requiere una estrategia que contemple restricciones en el orden en el que deben ejecutarse determinadas primitivas.

Orientaciones

Este momento es una buena oportunidad para reforzar el trabajo sobre estrategias de solución, valorando la división en subproblemas y la legibilidad de la solución, además de acompañar a las y los estudiantes durante la solución del desafío.



Cambiar todos los epigrafes por uno único que diga:
Solución posible al desafio ***Cargando los celus*** con procedimientos definidos para las distintas tareas.

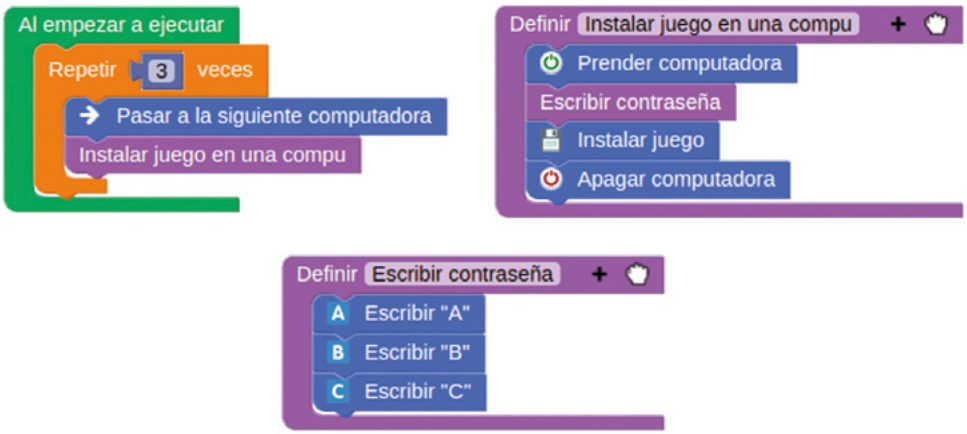
Cuando todos los grupos hayan finalizado el desafío, realizamos una breve puesta en común para que socialicen las estrategias de solución y, basándose en ellas, **reforzar la relación entre la estrategia de solución y la definición de procedimientos**[1] como una manera de mejorar la legibilidad para otras personas que interactúen con el programa. Aprovechamos las soluciones creadas para este desafío para **explicitar cómo**

[1] La noción de procedimiento se desarrolla en la secuencia "Procedimientos y repetición simple" de esta colección, que se encuentra disponible en el sitio curriculum.program.ar

las restricciones en el orden de uso de las primitivas (que vimos en el inicio) determinan la estrategia.

Como segunda parte de esta actividad, les proponemos a las y los estudiantes que ingresen al desafío de Pilas Bloques ***Instalando juegos***. Podemos aprovechar este desafío como una segunda instancia para que los grupos exploren el desafío y ganen autonomía en la construcción de sus programas, dado que podrán encontrar semejanzas con el desafío anterior.

En este desafío, las y los estudiantes se encontrarán con la particularidad que la secuencia de primitivas que se requiere para resolverlo es más larga en relación con el desafío anterior y, por lo tanto, definir un procedimiento para el ingreso de la contraseña aportará legibilidad a su solución. También, pueden identificar que el problema se resuelve repitiendo un proceso, y utilizar repeticiones y procedimientos para expresarlo.



Solución posible al desafío ***Instalando los juegos*** con un procedimiento para instalar el juego y otro para escribir la contraseña.

Cierre >

El **propósito de este momento** es conceptualizar determinadas restricciones en el orden de ejecución bajo la noción de estado y generalizar la noción a partir de ejemplos.

Orientaciones

Comenzamos por recuperar la experiencia del primer ejercicio, en el que las y los estudiantes corroboraron que no era posible ejecutar la primitiva Cargar celular si antes no habían recogido el cargador. Resaltamos esta situación en la que una acción requiere que otra se haya ejecutado previamente y conceptualizamos la noción de ***estado*** como la caracterización de un momento de la ejecución de un programa, prestando especial atención a qué acciones están restringidas o disponibles y qué continua-

ciones se habilitan en función de eso. En el ejemplo del desafío, podemos identificar dos estados: antes de haber recogido el cargador (estado "sin cargador") y después de recoger el cargador (estado "con cargador"). El primer estado restringe la utilización de la primitiva Cargar celular y la transición o cambio entre estos estados se produce cuando se ejecuta la primitiva Agarrar cargador.

Estas situaciones fuerzan a las y los programadores a proponer estrategias de solución que contemplen la solución de ciertos subproblemas en un orden determinado. Invitamos a identificar estados, transiciones entre estados y acciones habilitadas e inhabilitadas en artefactos computacionales con los que se interactúe en la vida cotidiana. Por ejemplo, en un ascensor automático podemos identificar dos estados: "puertas abiertas" y "puertas cerradas"; la transición entre ellos se produce con las acciones "cerrar puerta" y "abrir puerta" y en el estado "puertas abiertas" no se pueden realizar las acciones "subir" ni "bajar".

Actividad 2
El gran escape en yacaré

Esta actividad propone a las y los estudiantes resolver un desafío con varias etapas. Tanto para construir la solución como para revisarla y compartirla, será clave que elaboren una estrategia dividida en subproblemas y utilicen procedimientos para implementarla de manera legible.

Objetivos >

Se espera que las y los estudiantes:

- Propongan estrategias de solución a partir de la división en subproblemas.
- Implementen las estrategias de solución mediante programas legibles que utilicen procedimientos y repeticiones.

Inicio >

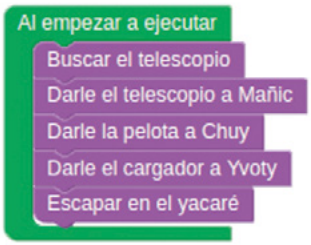
El **propósito de este momento** es recuperar la importancia de definir procedimientos para facilitar la comprensión de la estrategia de solución y ordenar la tarea de programación.

Orientaciones

Indicamos a los grupos de estudiantes que ingresen al desafío de Pilas Bloques ***El gran escape en yacaré***, analicen el problema y ensayen o especulen soluciones posibles.

Este desafío tiene la particularidad de ser un problema más extenso que los anteriores y que está formado por numerosas etapas que requieren distintas soluciones. Por eso, **a medida que las y los estudiantes ingresen al desafío y comiencen a interpretar y resolver el problema**, les recordamos que es clave crear una estrategia y los acompañamos a que identifiquen que en este caso la división en subproblemas provee una manera de resolver el desafío por partes y lograr una solución legible.

Podrán comenzar por identificar la secuencia de pasos que deben realizar para alcanzar el objetivo final (“Buscar telescopio”, “Darle el telescopio a Mañic”, “Darle la pelota a Chuy”, “Darle el cargador a Yvoty” y “Escapar en el yacaré”) y definir un procedimiento con una denominación representativa por cada uno.



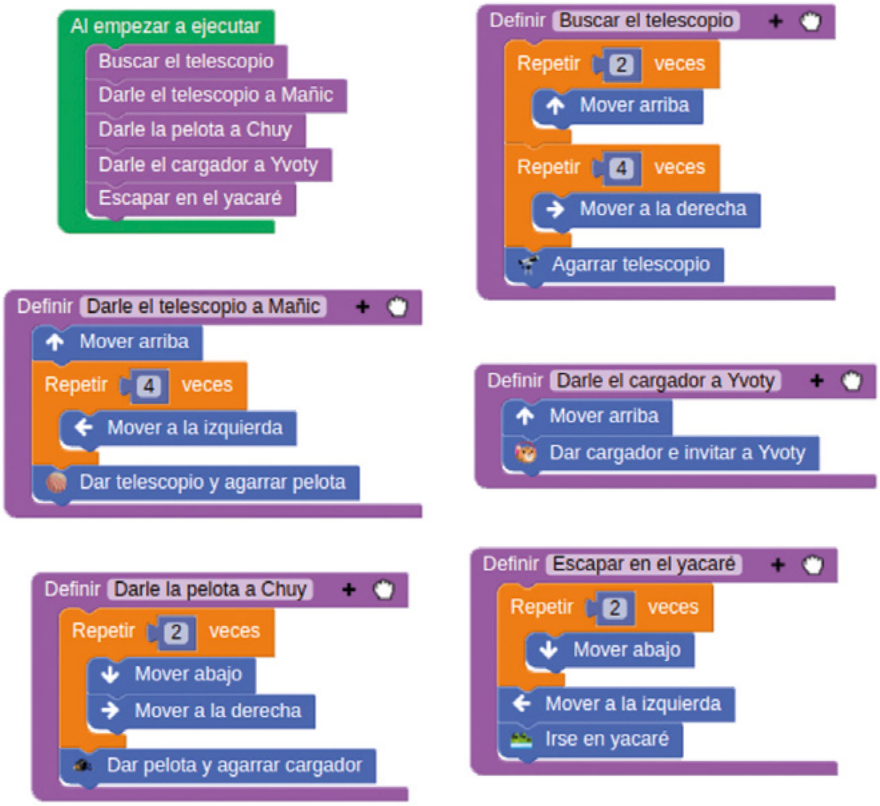
El programa principal de una solución posible en la que se definieron procedimientos para cada paso del desafío.

Desarrollo >

El **propósito de este momento** es completar y poner a prueba la solución del desafío.

Orientaciones

Los grupos concretan la estrategia de solución utilizando las primitivas disponibles y creando (y nombrando) los procedimientos.



Solución posible. Cada paso de la consigna está representada por un procedimiento que refleja la estrategia en el programa principal.

En un segundo nivel de detalle, es posible identificar subproblemas dentro de cada tarea y definir procedimientos para utilizarlos en la definición de procedimientos. En sintonía con esta idea, se pueden repensar las estrategias para subdividir pasos de la estrategia general, por ejemplo, en ir a un objeto o personaje y luego interactuar con él.

Una estrategia más refinada podría usar un procedimiento (Ir a Mañic) dentro de otro (Darle el telescopio a Mañic) para separar el problema de posicionarse en el personaje del de entregar el objeto.

La elaboración y la interpretación de este tipo de estrategias puede resultar demasiado abstracta para las y los estudiantes y, por lo tanto, lo consideramos como un refinamiento opcional y podemos proponerla y acompañar a los grupos que quieran seguir trabajando sobre el desafío.

Cierre >

El **propósito de este momento** es explicitar las ventajas de la legibilidad de las soluciones y explicitar las herramientas de programación utilizadas para conseguirla en diferentes casos.

Orientaciones

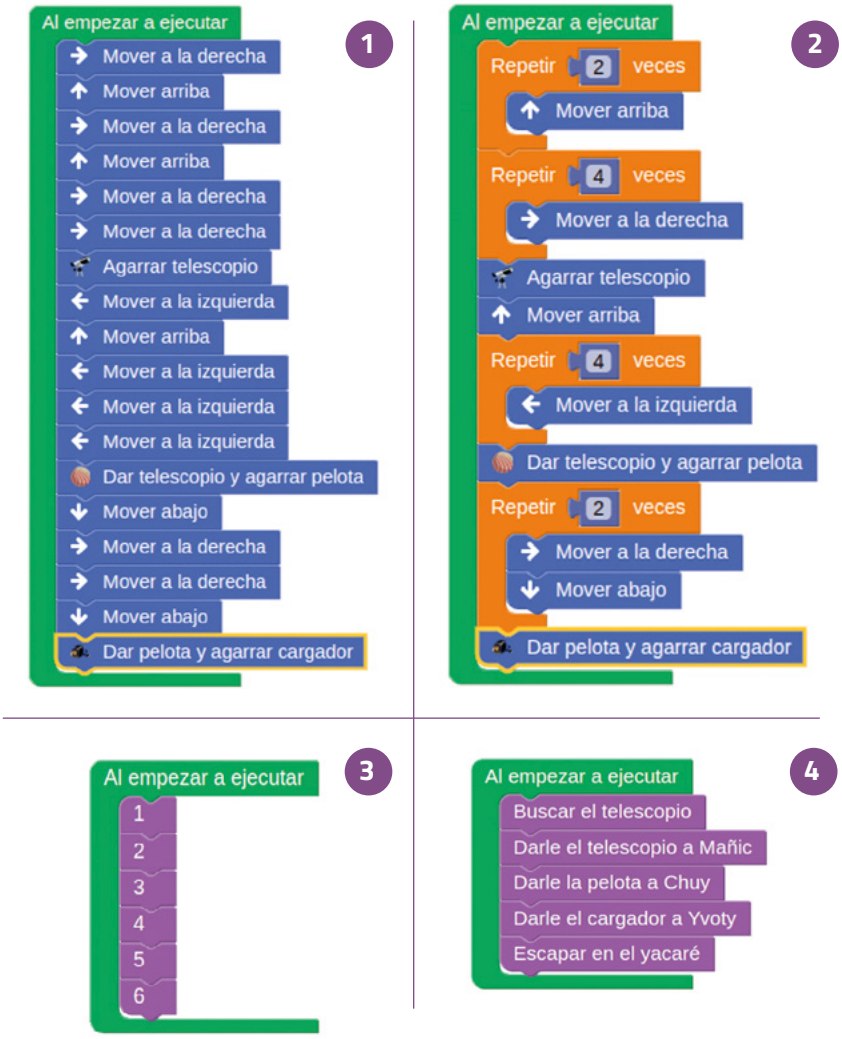
Se comienza con la presentación de los programas que realizó cada grupo a los demás. Se promueve que compartan cuál fue su razonamiento y por qué optaron por la estrategia realizada. Es importante mencionar que un mismo problema puede resolverse de diversas maneras y que promovemos el intercambio para conocer otras soluciones que pueden servirnos para abordar próximos desafíos, así como también identificar aspectos comunes en todas para reconocer aspectos generales del problema y su solución. El objetivo es lograr que los grupos puedan animarse a compartir sus experiencias[2].

¿En cuál o cuáles de las soluciones les resulta más fácil saber cuál es la estrategia de solución? ¿Cómo podría mejorarse? ¿En cuál o cuáles sería más sencillo corregir algún error o realizar una modificación específica? ¿Por qué? Si otro grupo tuviera que modificar su programa, por ejemplo, porque un personaje cambió de lugar, ¿qué podrían modificar previamente para facilitar la tarea?

Estas preguntas proponen que las y los estudiantes comparen soluciones con diferentes niveles de legibilidad. El objetivo es poner de manifiesto cómo, a medida que aumenta la complejidad del problema y la extensión

[2] De esta manera, queremos evitar estigmatizaciones o reforzar los estereotipos que hayan sido identificados por las y los estudiantes en otros encuentros.

del programa, la legibilidad resulta indispensable, porque permite encontrar errores fácilmente y comprender rápidamente cuál es la estrategia de solución.



Soluciones al desafío con legibilidad creciente. Comenzando por una solución que no utiliza ningún recurso para mejorar este aspecto, sigue el agregado de repeticiones, luego, la división en subproblemas y, finalmente, la utilización de nombres o denominaciones representativas para explicitar la estrategia de solución.

¿Qué herramientas de programación usaron para construir programas más legibles? ¿En qué casos utilizaron cada una?

A partir del intercambio, nos interesa asociar el uso de la repetición simple con expresar de manera legible la repetición de una o varias instrucciones una cantidad determinada de veces. También, afianzar el uso y definición de procedimientos y su denominación para dividir el problema en subproblemas y expresar la estrategia, indicando qué parte del problema resuelve cada uno.

Actividad 3
Limpiando el humedal

Las y los estudiantes integran los conceptos abordados (primitivas, procedimientos, repetición simple, división en subproblemas, legibilidad y estado) y refuerzan sus habilidades de programación en la solución de un desafío más complejo y con mayor autonomía.

Objetivos >

Se espera que las y los estudiantes:

- Propongan estrategias de solución a partir de la división en subproblemas.
- Implementen soluciones mediante programas legibles que utilicen procedimientos y repeticiones.

Inicio >

El **propósito de ese momento** es repasar la noción de estrategia de solución.

Orientaciones

Proponemos que los grupos ingresen al desafío de Pilas Bloques ***Limpiando el humedal***. Podemos aprovechar a recuperar algunas discusiones que tuvimos en la actividad anterior con respecto a pensar una estrategia de solución antes de programar, en especial, en desafíos extensos. Se los puede alentar a que lean con atención la consigna y observen el escenario para reconocer si es un problema que requerirá ser dividido en subproblemas o no.

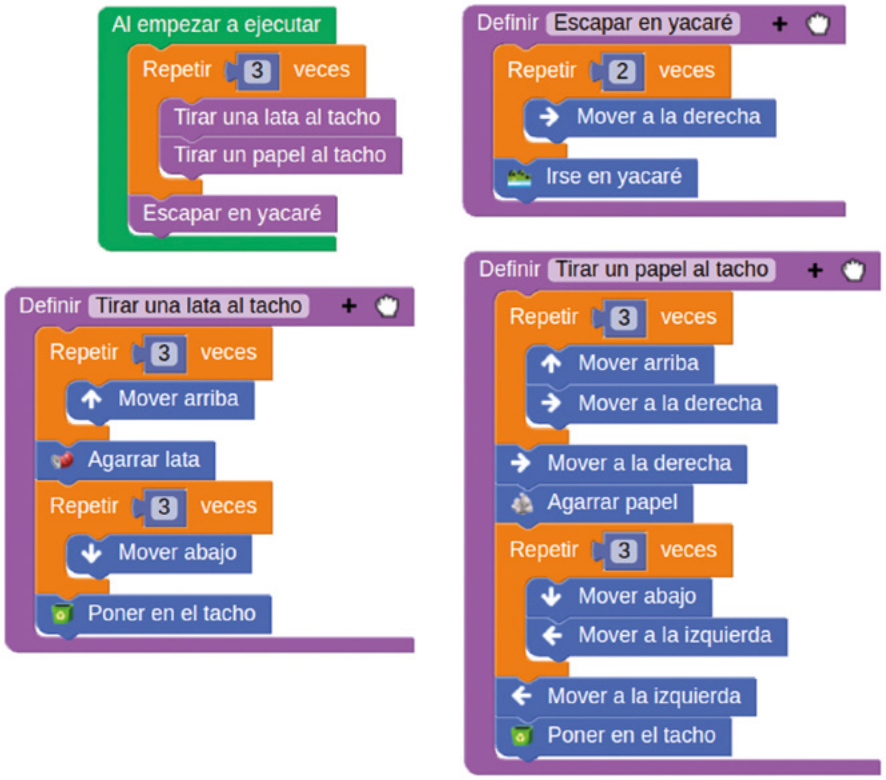
Desarrollo >

El **propósito de este momento** es proponer un desafío extenso que requiere que las y los estudiantes resuelvan autónomamente elaborando una estrategia de solución y un programa legible que integre primitivas, procedimientos, repeticiones y legibilidad.

Orientaciones

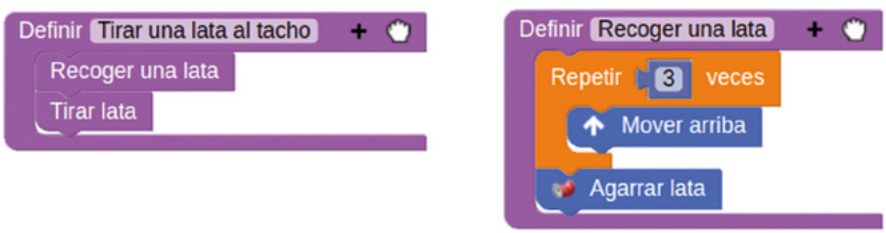
Al igual que el desafío anterior, este desafío puede resultar complicado de resolver si no se divide en subproblemas y difícil de lograr que la solución sea legible si no se usan repeticiones y denominaciones representativas de los procedimientos cuando es apropiado. En particular, la división en subproblemas permitirá estructurar la solución con procedimientos evitando que haya bloques de repetición anidados.

Como queremos que los grupos afiancen su autonomía, limitaremos la ayuda y las orientaciones en torno a recuperar la importancia de elaborar una estrategia de solución basada en la división en subproblemas como paso previo a la tarea de programación.



Una solución al desafío, con subproblemas diferenciados mediante el uso de procedimientos.

Al igual que en el desafío anterior, se pueden proponer divisiones dentro de los subproblemas que generan distintos niveles de **anidamiento** (es decir, procedimientos utilizados en la definición de otros procedimientos).



Ejemplo de solución con procedimientos anidados

Ejemplo de solución con procedimientos anidados

Cierre >

El **propósito de este momento** es repasar los conceptos y herramientas de programación abordados en los desafíos hasta el momento y generalizar su función a partir de las experiencias de resolver las actividades de la secuencia.

Orientaciones

¿Qué conceptos y herramientas de programación aparecieron en los desafíos que resolvieron? ¿Qué ejemplos pueden dar de sus programas? ¿Para qué utilizaron cada uno?

Recuperamos la experiencia de la actividad para que, entre todos los grupos, repasemos los conceptos involucrados en la solución y abordados en las secuencias.

- Las **primitivas** son comandos que vienen definidos y representan las únicas acciones que el autómata puede ejecutar. El resto de las acciones que queramos resolver debemos programarlas combinando las primitivas.
- La **estrategia** es el modo en el que decidimos resolver el problema. Si bien no involucra el uso de la computadora, es una parte fundamental del proceso de programación y está fuertemente asociada con el análisis y la comprensión del problema a resolver.
  - » Muchas veces, al analizar el problema a resolver para elaborar una estrategia, identificamos **subproblemas**, es decir, partes del problema que se pueden resolver por separado. Por ejemplo, en esta actividad, ir a buscar un papel o ir a buscar una lata.
  - » A veces, para **resolver un subproblema es necesario haber resuelto otros previamente**; de la misma manera, algunas primitivas requieren para ejecutarse que ya se hayan ejecutado otras. Esto induce la **noción de estado** como una caracterización del momento de ejecución (“ya se recogió el cargador”, “ya se juntaron todas las latas y los papeles”) que habilita determinadas acciones (si ya se recogió el cargador, se pueden cargar los celulares; si ya se recogieron todas las latas y los papeles, es posible irse en yacaré).

- Los **procedimientos** nos permiten reflejar en el programa la división en subproblemas.
  - » Nos permiten **crear comandos** (combinando primitivas), por ejemplo, definir el procedimiento Tirar una lata o Darle el telescopio a Mañic.
  - » Su uso **favorece la legibilidad**, porque hace más explícita la estrategia de solución. Sus nombres descriptivos (denominaciones representativas) y la posibilidad de aislar errores dentro de ellos agilizan el trabajo de programación e interpretación. Por ejemplo, para explicitar cada paso de ***El gran escape en yacaré***, o el orden en el que se recogen los papeles y las latas en ***Limpiando el humedal***.
  - » Nos permiten **reutilizar** la solución de subproblemas que aparecen varias veces dentro del mismo problema, por ejemplo, al definir el procedimiento Instalar juegos en una compu o Nuevos comandos (de desafíos anteriores).
- Las **repeticiones** ayudan a resolver ocurrencias consecutivas de un mismo subproblema o problema. Para esto es importante haber identificado **patrones** dentro del desafío. Hay que tener en cuenta que pueden producirse casos de borde. Por ejemplo, en el desafío ***Instalando juegos***, identificamos que la secuencia de prender la computadora, escribir la contraseña, instalar el juego y apagar la computadora se repetía. Por eso, pudimos usar una repetición para aprovechar este patrón del problema y escribir un programa más legible.
- La **legibilidad**, es decir, la facilidad de comprensión de los programas por otras personas es un aspecto fundamental en la tarea de programación. La forma de lograr legibilidad en los programas es a través del uso de **procedimientos** para dividir el problema en subproblemas más pequeños, utilizar **denominaciones representativas** y utilizar repeticiones cuando se pueda. La importancia de la legibilidad la abordamos en el cierre de la **Actividad 2**, cuando comparamos las diferentes versiones de la solución (con y sin repeticiones y procedimientos).

Secuencia didáctica 7

Creamos desafíos de procedimientos

Procedimientos

¿Cuándo es conveniente usar procedimientos para resolver un problema? ¿Cómo es un desafío de Pilas Bloques para trabajar procedimientos en el aula?

En esta secuencia, las y los estudiantes crean desafíos en Pilas Bloques que, para ser resueltos, requieren el uso de procedimientos. Así revisitan los conceptos de estrategia y procedimientos, e identifican relaciones entre la aplicación de estas nociones y las características de un problema. La creación de desafíos funciona como una actividad con un objetivo conceptual concreto que apela a la creatividad de las y los estudiantes.

Actividad

Las y los estudiantes crean desafíos que requieran el uso de procedimientos para su solución, para reforzar su conceptualización e identificar qué características de una situación problemática están asociadas a la utilización de esta herramienta.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Soluciones a problemas computacionales

- Diseño de programas: estrategias de solución.

Eje: Lenguajes de programación

- Herramientas de lenguaje de programación: procedimientos, repetición simple.

Objetivos de aprendizaje

- Identificar, de manera más general o abstracta, relaciones entre la estructura de un problema y la necesidad de utilizar procedimientos en un programa que lo resuelva, especialmente para mejorar la reutilización y la legibilidad del programa.

Saberes previos de CC

Área: Programación

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategias de solución.

Eje: Lenguajes de programación

- Herramientas de lenguaje de programación: procedimientos, repetición simple.

Materiales necesarios

- Dispositivos con Pilas Bloques instalado o acceso a su versión en línea https://pilasbloques.program.ar/

Actividad
Nuestro desafío de Pilas Bloques para trabajar procedimientos

Objetivos >

Se espera que las y los estudiantes diseñen y creen un desafío cuyo objetivo sea abordar el uso de procedimientos[1].

Inicio >

El **propósito de este momento** es explicitar que existe una relación entre el diseño de un desafío de Pilas Bloques (en particular, el escenario y los bloques disponibles) y las nociones de programación que se propone abordar.

Orientaciones

Para comenzar la clase, evocamos alguna experiencia previa de las y los estudiantes con Pilas Bloques para identificar, en un desafío puntual, características del diseño del desafío que requieren la aplicación de una herramienta o noción particular. Por ejemplo, si ya realizamos la secuencia "Creamos desafíos de repetición. Repetición simple" de esta colección, podemos recuperar los desafíos creados por las y los estudiantes para identificar las decisiones de diseño que tomaron para que fuera necesario el uso de la repetición. O bien, podemos comenzar mostrando a las y los estudiantes el siguiente escenario de un desafío construido con el Creador de desafíos de Pilas Bloques para que hipoteticen cuál es el problema y una solución con los bloques que consideren que serían necesarios.

[1] Esta secuencia requiere que las y los estudiantes estén familiarizados con el funcionamiento del Creador de desafíos de Pilas Bloques. Para ello, se recomienda trabajar previamente con, al menos, la **Actividad 1** de la secuencia "Creamos desafíos de repetición. Repetición simple" de esta colección, que se encuentra disponible en el sitio curriculum.program.ar.

¿Cuál dirían que es el objetivo de resolver este desafío en clase? ¿Cuál dirían que es "el tema" que se aborda con este desafío? ¿Qué bloque les parece que los docentes queremos que ustedes apliquen en sus soluciones? ¿Qué decisiones de diseño del desafío hacen que esto sea así? ¿Qué cosas podrían cambiar y cuáles no para que siga siendo un desafío que requiere el bloque mencionado?

A partir de estas preguntas y el debate sobre las respuestas, interesa que las y los estudiantes identifiquen que:

- Un bloque clave para resolver este desafío es el bloque de repetición.
- Quienes diseñaron el desafío tomaron decisiones para que se requiera usar el bloque de repetición: decidieron colocar cinco objetos, uno al lado del otro, para que se requiera repetir cinco veces una misma acción.
- Podríamos crear otro desafío con el mismo objetivo. Solo hay que conservar la intención de que la solución requiera la repetición de una acción una suficiente cantidad de veces como para que no sea eficiente replicar los bloques tantas veces como se deba realizar la acción.

Desarrollo >

El **propósito de este momento** es promover que las y los estudiantes reflexionen sobre qué características de un desafío de Pilas Bloques requieren el uso de procedimientos. Para ello, deberán crear un desafío que requiera necesariamente utilizar en su solución esta herramienta.

Material de referencia para docentes

¿Podemos aprender a programar creando desafíos de programación?

Para resolver un desafío de Pilas Bloques, las y los estudiantes tienen que, a partir de un escenario y una consigna, construir un programa (esto involucra, en particular, definir una estrategia y utilizar determinadas herramientas de programación para implementarla). Al hacerlo, como docentes, nos interesa que asocien características de un problema de un nivel más general (por

ejemplo, tiene partes claramente identificables o implica la repetición de una tarea puntual) con necesidades de un programa (contar con procedimientos definidos con nombres representativos o utilizar un bloque de repetición), con la intención de que puedan resolver situaciones problemáticas más generales que las de los desafíos puntuales que les presentamos.

En línea con el esfuerzo de reconocer generalidades en los problemas como una manera de avanzar en la construcción de los programas que los resuelven, proponemos invertir el camino: a partir de una característica de un programa, las y los estudiantes deberán proponer un desafío que se resuelva con un programa con esta característica. Si antes nos podíamos preguntar "¿**Necesitaremos procedimientos** para resolver este problema?", ahora debemos respondernos "¿**Cómo debe ser un problema** para que su solución involucre el uso de procedimientos?". La respuesta a ambas preguntas implica una comprensión profunda tanto de la herramienta de programación como de su aplicación en la solución de situaciones problemáticas.

Orientaciones

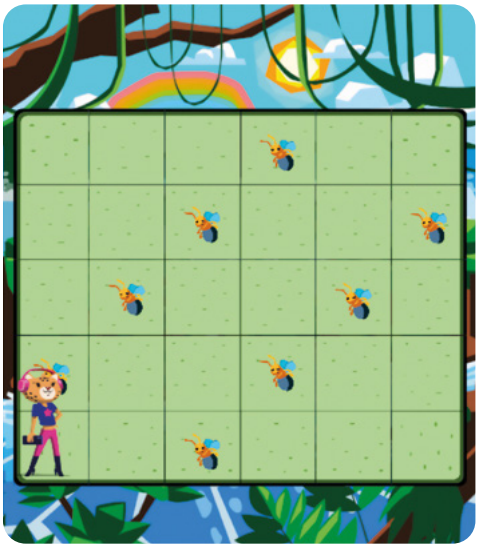
Organizamos la clase en grupos heterogéneos pequeños[2]. Les entregamos a cada uno una tarjeta con una consigna (Ver Anexo). Pueden elegir el personaje de su preferencia. Anticipamos que no deben mostrar la tarjeta a los demás grupos y que la dinámica de la actividad consistirá en que otro grupo resuelva el desafío que crearán.

Tarjetas con las consignas de la actividad.

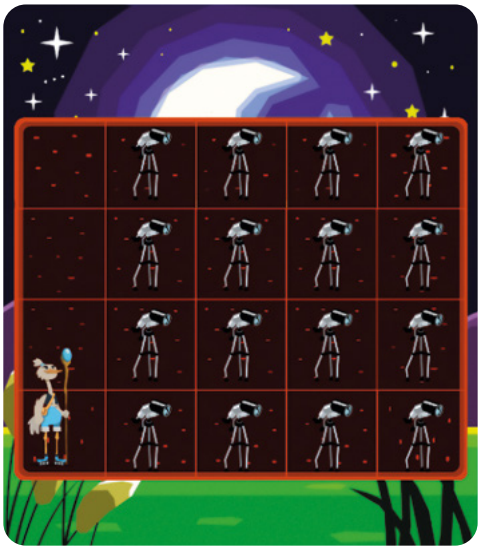
[2] Se pueden consultar dinámicas lúdicas para el armado de grupo heterogéneos en "Enseñar computación desde la mirada de la Educación Sexual Integral (ESI)", 10.

Con esta actividad buscamos promover en las y los estudiantes una reflexión más profunda sobre el uso de procedimientos, partiendo de preguntarse qué características debe tener un desafío para que sea conveniente utilizar procedimientos para resolverlo de la manera indicada en cada tarjeta.

La tarjeta que requiere que se use **dos veces un mismo procedimiento** apunta al **uso de los procedimientos para favorecer la reutilización de una solución**. Esto es posible cuando un subproblema aparece varias veces en el problema general y es posible definir un único procedimiento y utilizarlo cada vez que aparece el subproblema, por ejemplo, esto es lo que sucede en los desafíos de Pilas Bloques ***Nuevos comandos*** e ***Yvoty y las luciérnagas***.



La tarjeta que requiere que se use **un procedimiento dentro de una repetición** apunta al uso de los **procedimientos para expresar tareas que son más complejas que los comandos primitivos y favorecer la legibilidad del programa**. Esto cobra sentido cuando el problema requiere resolver un subproblema (es decir, una tarea lo suficientemente compleja como para considerarla un problema en sí mismo) que aparece repetido varias veces, una a continuación de la otra. Por ejemplo, esto es lo que sucede en los desafíos de Pilas Bloques ***Reparadora de telescopios*** o ***Instalando juegos***.



Teniendo en cuenta estas ideas, invitamos a los grupos a que ingresen al Creador de desafíos de Pilas Bloques y comiencen a pensar cómo será el desafío que van a crear. Recorremos los grupos prestando atención a las inquietudes y los avances. Es probable que necesitemos acompañar el trabajo de cada grupo individualmente, especialmente para ayudarlos a identificar **cuáles son las características clave que deben incorporar al desafío** para cumplir con el objetivo pedido. Algunas estrategias que podemos usar son las siguientes.

- Sugerir que **no diseñen escenarios complejos** (para que la creación y la solución del desafío requieran el menor tiempo posible).
- Acompañar a cada grupo con preguntas orientadoras para que **reflexionen sobre cómo deben ser los escenarios** de los desafíos para cumplir con los requisitos de la tarjeta.
- Proponer a los grupos que no sepan cómo comenzar que **escriban primero un boceto de la forma del programa** que cumpla con el objetivo de la tarjeta para avanzar hacia una solución abierta, pero más concreta sobre la que seguir trabajando.
- Proponer a los grupos que identifiquen y consulten como modelo **desafíos que ya hayan resuelto** con programas similares que cumplan con los requisitos de la tarjeta que les tocó.

Para poner a prueba los desafíos, organizamos el **intercambio de los desafíos entre los grupos para que sean resueltos** por un grupo diferente al que los creó y que hayan trabajado con una tarjeta diferente[3].

[3] Ver opciones para gestionar el intercambio de desafíos y soluciones en la **Actividad 1** de la secuencia "Creamos desafíos de repetición. Repetición simple".

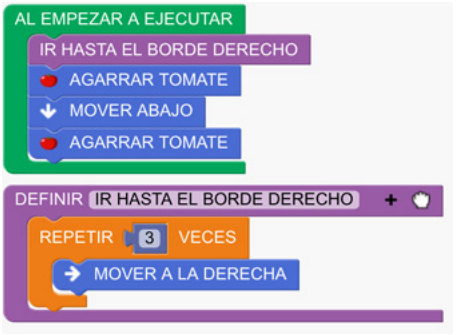
Si se detectan dificultades en el diseño mientras se acompaña el trabajo de los grupos, también se puede **realizar un intercambio de desafíos previo a su finalización como una iteración que sea parte del proceso de creación**. Por eso, es importante que los grupos arriben rápido a una primera versión del desafío que se pueda probar (por el mismo grupo creador o por otro). Para ello, se sugiere no diseñar escenarios muy complejos. Recomendamos a los grupos creadores que tomen nota de los cambios que realicen en los desafíos, porque los retomaremos en el intercambio que haremos al cierre de la actividad.

A medida que los grupos resuelven los desafíos creados por otros, promovemos la devolución al grupo creador para motivar reflexiones que permitan asociar la disposición del escenario y los bloques disponibles del desafío con las soluciones posibles. Podemos hacer preguntas que los inviten a hipotetizar variaciones del desafío, identificar dificultades o errores y explicitar los razonamientos que hicieron en el proceso de creación.

¿Todas las soluciones son válidas? ¿Cuándo está cumplido el objetivo?

Es probable que surjan desafíos que admitan algunas soluciones que cumplen con los requerimientos de las tarjetas y otras que no. Por ejemplo, para la tarjeta que requiere dos veces el uso de un procedimiento, podría plantearse el siguiente escenario con la solución esperada por el grupo que lo definió.

Sin embargo, también es válida una solución que no cumple con el objetivo de la tarjeta.



Un programa que resuelve el deasfío pero no cumple el objetivo de usar dos veces un procedimiento.

Descubrir esta solución invita a volver a pensar sobre la disposición del escenario y, por lo tanto, refuerza las reflexiones que apuntan a asociar características del problema con características del programa. Estas sutilezas, que pueden parecer detalles engorrosos o innecesarios, son valiosas para que se establezca el ida y vuelta entre los grupos y se generen sucesivas instancias de análisis, argumentaciones, ensayo y puesta a prueba.

En el último ejemplo, podemos identificar que el problema es que se puede pasar de un tomate a otro sin necesidad de regresar al borde y avanzar nuevamente hacia la derecha (que es lo que plantea el procedimiento ir hasta el borde derecho). Este inconveniente se puede resolver impidiendo este recorrido.

Un escenario que requiere utilizar un procedimiento dos veces.

Cierre >

El **propósito de este momento** es abrir una instancia de metacognición para identificar que, en el proceso de construcción de los desafíos, fue necesario descubrir relaciones entre la disposición del escenario y los bloques disponibles con las características del programa que lo resuelve; y también para explicitar estas relaciones y generalizarlas, teniendo en cuenta la importancia del análisis del problema a la hora de proponer un programa para solucionarlo.

Orientaciones

El objetivo de esta instancia es construir un espacio de debate y análisis, atendiendo a que ningún grupo se sienta expuesto o atacado ni se menosprecie el trabajo de nadie. Invitamos a los grupos a que compartan sus desafíos y que relaten las decisiones que tomaron y el proceso iterativo que realizaron con el grupo que lo probó.

De esta experiencia nos interesa señalar que **la complejidad de la actividad reside en encontrar una relación entre la forma del desafío (el escenario y los bloques disponibles) y las soluciones posibles**. Es importante rescatar cómo cada grupo fue construyendo esta relación[4]. Podemos mencionar que este ejercicio es similar al que hacen cuando resuelven un desafío ya creado (es decir, buscar relaciones entre el problema que tienen que resolver y las herramientas de programación que van a utilizar para resolverlo), pero que hacerlo "al revés" permitió concentrarse en este aspecto y, por lo tanto, forma parte de una instancia más profunda de su aprendizaje de programación.

Para explicitar las relaciones encontradas a partir de cada tarjeta e intentar expresarlas de manera más general, podemos partir de la comparación de desafíos que cumplan con el objetivo con otros que no. Podemos considerar tanto las versiones finales de los desafíos como las intermedias (lo que nos permite recuperar los motivos por los cuales los desafíos tuvieron que ser mejorados) o desafíos de ejemplo que no formen parte de la producción de los grupos.

Para la tarjeta que requiere usar dos veces el mismo procedimiento

Buscamos que las y los estudiantes lleguen a una descripción que relacione el escenario con el programa, por ejemplo: "Utilizamos dos veces el mismo procedimiento cuando el escenario tiene dos partes iguales, pero

[4] Si se había trabajado previamente con la secuencia "Creamos desafíos de repetición. Repetición simple" de esta colección, podemos recuperar ese intercambio de cierre para que las y los estudiantes retomen las ideas centrales y las reconozcan en esta nueva experiencia.

separadas", "Si el escenario tiene un patrón de objetos en dos lugares diferentes, conviene usar dos veces el mismo procedimiento para resolverlo".

Además o en lugar de los desafíos creados por los grupos, podemos citar los siguientes ejemplos e identificar cuáles cumplen con el objetivo y cuáles no.

Dos pares de desafíos para comparar y contrastar. Los de la izquierda no cumplen con el objetivo (el de arriba, porque los subproblemas no están separados; y el de abajo, porque los subproblemas son todos distintos), mientras que los de la derecha sí.

Para arribar a la idea de que si un subproblema (en este caso, expresado como un patrón en el escenario) aparece repetidas veces, podemos construir un procedimiento para resolverlo y utilizarlo todas las veces que sea necesario, podemos preguntar: *¿Cómo podríamos cambiar estos desafíos para que requieran usar tres veces el mismo procedimiento? ¿Y seis?*

Dos desafíos de Pilas Bloques creados por estudiantes en los que un mismo patrón aparece dos veces separadas en el escenario.

Para la tarjeta que requiere usar un procedimiento dentro de una repetición

Podemos comparar y contrastar los ejemplos del desarrollo de la secuencia con los siguientes.

Dos versiones de un desafío hecho con el Creador, una que no cumple con el objetivo (izquierda) y otro que sí (derecha).

Dos desafíos existentes en Pilas Bloques que cumplen con el objetivo de las tarjetas.

En todos estos ejemplos, existe un subproblema que se repite exactamente igual, varias veces y uno a continuación del otro. Para reforzar la posibilidad de reconocer esto, podemos comparar las dos versiones del desafío de ejemplo del trofeo, en el que, dado que en la primera no aparece dos veces una pelota seguida de un trofeo, no es posible repetir el procedimiento que resuelve ambos objetos.

Buscamos que arriben a una conclusión similar a "Utilizamos un procedimiento dentro de una repetición cuando debemos realizar una acción compleja varias veces seguidas" o "Es conveniente repetir un procedimiento cuando encontramos un patrón uno a continuación del otro".

Dos producciones reales realizadas por estudiantes que repiten procedimientos. A la izquierda, se repite tres veces el procedimiento "Comer churrasco de abajo", para resolver la primera columna. A la derecha, el desafío es similar, pero su solución es más larga.

Para generalizar esta situación, podemos preguntar: *¿Cuándo podría ser necesario utilizar dos procedimientos dentro de una repetición?* Buscamos arribar a que el subproblema que se repite está formado por dos partes claramente diferenciadas (por ejemplo, una fila de latas debajo de una fila de papeles) y asociar **la repetición en el programa** con **la repetición de un subproblema** (un patrón en el escenario) y **el uso del procedimiento dentro de esa repetición** para **expresar la solución al subproblema** (que puede resolverse con un procedimiento o con más).

Anexo
Tarjetas de objetivo

Creá un desafío para que se resuelva con un programa en el que aparece dos veces un mismo procedimiento.

Creá un desafío para que se resuelva con un programa que usa un procedimiento dentro de una repetición.

Secuencias didácticas

> Unidad 2:

Alternativa condicional

8. **¿Cómo se resuelven problemas cambiantes?** Estrategia y alternativa condicional
9. **Resolvemos recorridos cambiantes**. Alternativa condicional y repetición
10. **Programamos estrategias para problemas cambiantes**. Estrategia, alternativa condicional y repetición
11. **Creamos desafíos cambiantes**. Estrategia, repetición y alternativa condicional

Secuencia didáctica 8

¿Cómo se resuelven problemas cambiantes?

Estrategia y alternativa condicional

¿Es posible resolver un desafío con escenarios que varían con un único programa? ¿Cómo cambia la estrategia en estas situaciones?

En esta secuencia se presentará la **alternativa condicional**, una herramienta de programación que resuelve la necesidad de ejecutar instrucciones solo en determinados casos. Esto permite construir un solo programa que pueda ser usado en diferentes escenarios.

Actividad 1

A partir de los desafíos de Pilas Bloques ***La pelota indecisa*** y ***¿Pelota o paleta?***, las y los estudiantes conocen dos comandos de alternativa condicional.

Actividad 2

A partir de los desafíos de Pilas Bloques ***Las estrellas de Mañic*** y ***La estrella especial***, las y los estudiantes elaboran estrategias que involucran la ejecución de un curso de acción u otro dependiente de una condición.

Actividad 3

A partir de los desafíos de Pilas Bloques ***Hilera de latas*** y ***Turistas latosos***, las y los estudiantes elaboran estrategias que involucran la ejecución adicional de acciones dependiendo de una condición.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: procedimientos, alternativa condicional.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad.

Objetivos de aprendizaje

- Incorporar la alternativa condicional como una herramienta para la solución de problemas computacionales que involucren distintos escenarios.
- Diferenciar características fijas y variables de una situación problemática y organizar la estrategia de solución considerando las características fijas y variables.
- Reconocer la existencia de soluciones previas que pueden ser utilizadas como subtareas en una nueva solución.

Saberes previos de CC

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, secuencias, procedimientos, repeticiones.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad.

Materiales necesarios

- Dispositivos con Pilas Bloques instalado o acceso a su versión en línea https://pilasbloques.program.ar/

Actividad 1
La pelota indecisa + ¿Pelota o paleta?

En los desafíos de Pilas Bloques ***La pelota indecisa*** y ***¿Pelota o paleta?***, se presentan problemas con escenarios variables para introducir la noción de alternativa condicional y los bloques correspondientes.

Objetivos >

Se espera que las y los estudiantes conozcan la alternativa condicional como herramienta para abordar problemas computacionales variables y la utilicen para resolver desafíos de Pilas Bloques.

Inicio >

El **propósito de este momento** es introducir la idea de que un problema se puede presentar en escenarios diferentes y, aun así, es posible encontrar un patrón y diseñar una estrategia de solución para resolverlo construyendo un único programa.

Orientaciones

Organizamos la clase en grupos heterogéneos pequeños[1]. Invitamos a los grupos a que ingresen al desafío de Pilas Bloques ***La pelota indecisa***. En su exploración, es fundamental para esta secuencia que las y los estudiantes adviertan que, al presionar el botón Ejecutar, el escenario cambia.

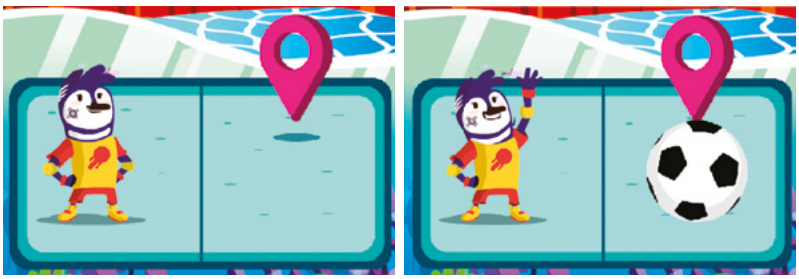
Al ingresar al desafío, se verá un mensaje que advierte a las y los estudiantes que se presentarán diferentes escenarios al ejecutar el programa.

[1] Se pueden consultar dinámicas lúdicas para el armado de grupos heterogéneos en "Enseñar computación desde la mirada de la Educación Sexual Integral (ESI)", 10.

¿Cuál dirían que es la diferencia más importante entre este desafío y los que ya trabajamos? ¿Cuáles son los escenarios de este desafío? ¿Cómo lo podemos averiguar? ¿Cómo describirían los escenarios? ¿Qué cambia y qué se mantiene igual entre ellos?

En este desafío, las y los estudiantes se encontrarán (por primera vez) con un desafío con dos posibles escenarios: mientras que la disposición de los casilleros es igual en ambos, en uno, el casillero de la derecha está vacío, y, en el otro, hay una pelota de fútbol para patear.

Es importante dejar en claro que el objetivo es elaborar **un único programa que resuelva todos los escenarios posibles del desafío**.



Los dos escenarios posibles del desafío ***La pelota indecisa***.

Desarrollo >

El **propósito de este momento** es que las y los estudiantes indaguen sobre el uso y la función de los bloques de alternativa condicional y los sensores lógicos.

Orientaciones

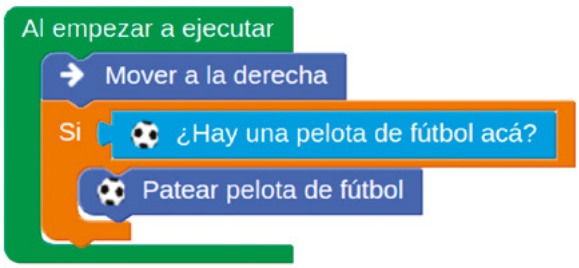
Una vez reconocidos los escenarios posibles, reforzamos la consigna (elaborar un **único programa** que resuelva **todos los escenarios posibles**) e invitamos a los grupos a que intenten resolver el desafío.

¿Es posible resolver este desafío con las herramientas que conocen un desafío en el que no sepamos cómo va a ser el escenario? ¿Por qué? ¿Qué bloques nuevos aparecen? ¿Cómo funcionan?

Para construir la solución, es fundamental que los grupos exploren los bloques de las categorías que aparecen por primera vez en este desafío: **alternativas** y **sensores**. Alentamos a que combinen estos bloques y exploren cómo usarlos en sus programas para construir un programa que funcione en ambos escenarios.

En el menú lateral, debajo de las categorías habituales, se encuentran Alternativas y Sensores.

Dado que es la primera vez que se encuentran con bloques de estas categorías, es probable que debamos guiarlos en esta ocasión. También, cuando lo consideremos pertinente, realizaremos una puesta en común para que quienes que hayan comprendido el funcionamiento de estos nuevos bloques compartan lo visto con el resto de la clase.



Una solución posible al desafío.

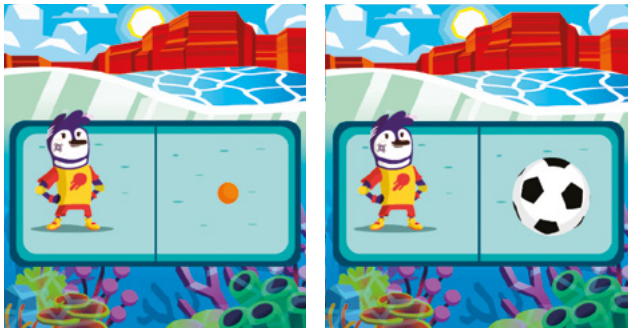
Cuando todos los grupos hayan logrado que el programa construido funcione en los dos escenarios posibles, reforzamos la idea de que el objetivo es lograr resolver ambos escenarios con un solo programa. Por lo tanto, el programa debe atender las dos alternativas y no es suficiente con atender a una sola. Convocamos a una charla para analizar los nuevos bloques que necesitaron y utilizaron: el bloque de alternativa si y el sensor ¿Hay una pelota de fútbol acá?

¿Qué categorías nuevas de bloques encontraron? ¿Qué bloques utilizaron para resolver el desafío? ¿Cómo funciona cada uno?

A partir de las respuestas, construimos la siguiente conclusión: todo lo que pongamos dentro del bloque si se ejecutará solamente si la respuesta a la pregunta que formula el sensor es afirmativa. **Por primera vez, están abordando un desafío en el que existe una parte del programa que, según el escenario, se ejecutará o no**. Si no hay pelota de fútbol, la instrucción Patear pelota no se ejecutará y el programa culminará (dado que no hay otras instrucciones por ejecutar).

Finalizada esta instancia, proponemos a los grupos continuar con el desafío de Pilas Bloques ***¿Pelota o Paleta?*** En este ejercicio, el uso de la alternativa condicional se complejiza. En la categoría Alternativas, aparece un **nuevo bloque**: si, si no. Es importante que tengamos en cuenta que algunos grupos podrán resolver el desafío sin utilizar el nuevo bloque usando dos veces el bloque si. Estos programas son un insumo valioso para las comparaciones con las soluciones que sí lo usan.

Anticipamos que el desafío tiene múltiples escenarios y recordamos que es necesario ejecutar varias veces el programa para relevar las variaciones del escenario.



Escenarios posibles del desafío ***¿Pelota o Paleta?***

Cierre >

El **propósito de este momento** es conceptualizar el funcionamiento de los bloques si y si, si no y la noción de condición.

Orientaciones

Invitamos a los grupos a que compartan sus estrategias de solución y sus programas. Buscamos identificar **los tipos de bloques utilizados y su combinación** para recuperar la idea de que es posible construir diferentes soluciones. Aprovechamos el intercambio para, principalmente, analizar el funcionamiento de los bloques de alternativa condicional. Si lo consideramos necesario, podemos invitar explícitamente a explorar el comportamiento del nuevo bloque (si, si no) a los grupos que no lo hayan utilizado.

Tres soluciones que reflejan combinaciones posibles de bloques de alternativa condicional para resolver el desafío.

Como conclusión del intercambio sobre lo trabajado, nos centramos en las siguientes ideas.

- Las soluciones tienen una **condición**, es decir, una pregunta cuya respuesta es inequívocamente sí o no.
- Para construir la condición, utilizamos bloques de la **categoría Sensor**, que son bloques que nos permiten introducir una pregunta para obtener información del escenario.
- Los comandos dentro del bloque si se ejecutarán solamente si se cumple la **condición**. Si no se cumple, no se ejecutarán; el programa seguirá y se ejecutará el siguiente bloque debajo del bloque de alternativa.
- En cambio, con el bloque si, si no siempre se ejecutarán comandos. Cuáles se ejecute dependerá de si se cumple o no la **condición** (es decir, si la respuesta a la pregunta de la condición es sí o no). Luego el programa seguirá su ejecución fuera del bloque de alternativa.

Para reforzar esta conceptualización, proponemos identificar situaciones de la vida cotidiana en la que realizamos determinadas acciones según si se cumple o no una condición, por ejemplo: "Si llueve llevo paraguas, si no llevo gorra" o "Si me acompañás, vamos a la plaza".

Material de referencia para docentes

Actividades desenchufadas para reforzar el uso de condicionales

Para reforzar lo visto, podemos realizar actividades desenchufadas para que las y los estudiantes analicen situaciones cotidianas que involucran condiciones.

- "Actividad 1. La carrera de los palos españoles", de "Condicionales", en Areces, C., Benotti, L., Cortez, J. J., *et al.* (2018). *Ciencias de la computación para el aula: 2.do ciclo de primaria: libro para docentes*. Ciudad Autónoma de Buenos Aires: Fundación Sadosky. Disponible en: ***https://repositorio.curriculum.program.ar/repositorio-diseno-curricular/alternativa-condicional/***
- "Actividad 2. ¿Qué harías si…?", de "A veces sí, a veces no", en Czemerinski, H., Dabbah, J., Floris C. R., *et al.* (2018). *Ciencias de la computación para el aula: 1.er ciclo de primaria: libro para docentes*. Fundación Sadosky. Disponible en: ***https://repositorio.curriculum.program.ar/repositorio-diseno-curricular/a-veces-si-a-veces-no/***

Actividad 2
Las estrellas de Mañic + La estrella especial

Las y los estudiantes diseñan estrategias de solución que utilicen alternativa condicional para resolver desafíos de Pilas Bloques con escenarios variables (***Las estrellas de Mañic*** y ***La estrella especial***).

Objetivos >

Se espera que las y los estudiantes:

- Elaboren estrategias de solución para problemas variables.
- Incorporen la alternativa condicional a la formulación de estrategias.

Inicio >

El **propósito de este momento** es que las y los estudiantes elaboren una **estrategia de solución** que incluya alternativa condicional a partir de la exploración y el análisis de los escenarios del desafío.

Orientaciones

Proponemos a los grupos que exploren el desafío de Pilas Bloques ***Las estrellas de Mañic*** y los bloques disponibles. Enfatizamos la importancia de explorar todos los escenarios cliqueando el botón Ejecutar. Recuperamos que, al igual que en las secuencias anteriores, un primer paso para la resolución de los desafíos consiste en analizar el problema para diseñar una estrategia de solución. Podemos alentar a que reconozcan similitudes con un desafío que resolvieron anteriormente (***¿Pelota o paleta?***): el escenario cambia con cada ejecución (por lo que se presentan diferentes casos) y esto requiere contar con bloques de alternativa y sensores.

Observen todos los escenarios del desafío. ¿Cuántos son? ¿Cómo los describirían? ¿Qué se mantiene y qué cambia en cada caso?

En ***Las estrellas de Mañic***, la tranquera puede aparecer en dos ubicaciones diferentes.

Como primer paso para la elaboración de la estrategia, proponemos **analizar los escenarios para identificar características que varían y otras que permanecen constantes**. En este caso, la disposición de los casilleros y de la estrella se mantienen, pero cambia la ubicación de la tranquera.

¿Cómo podemos saber qué camino debe tomar Mañic? ¿En qué lugar debe estar para saber por dónde seguir? ¿Qué partes o subproblemas identifican en el recorrido? ¿Cómo sería una estrategia de solución para este problema?

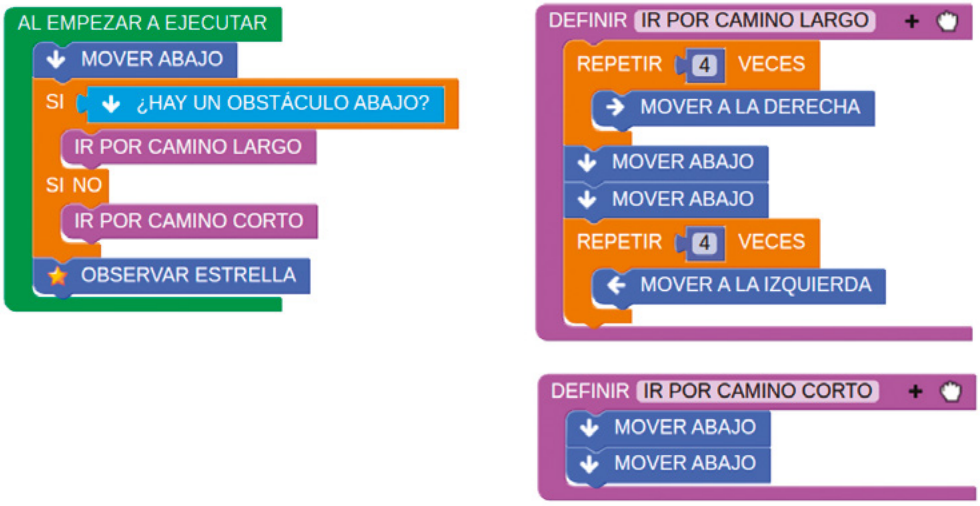
Recorremos los grupos de trabajo para acompañar a las y los estudiantes a elaborar una estrategia que divida el desafío en subproblemas: avanzar hasta el casillero en el que pueden detectar dónde está la tranquera, avanzar por camino libre y observar la estrella.

Desarrollo >

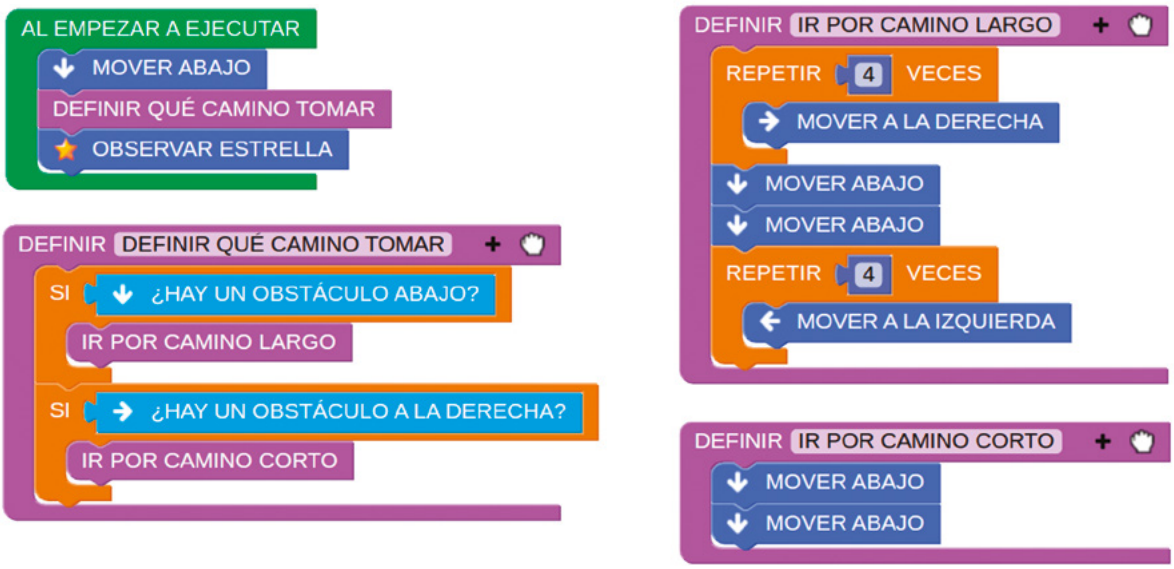
El **propósito de este momento** es reforzar la importancia de identificar las características constantes y variables en un problema para plantear estrategias de solución que puedan atender todos los escenarios posibles y que estas estrategias (o parte de ellas) puedan reutilizarse en otros problemas.

Orientaciones

Damos inicio a la programación de las estrategias de solución por parte de los grupos. Con este desafío, podemos evaluar si los grupos han comprendido cómo utilizar la alternativa condicional en la implementación de su estrategia. Es posible que algunos grupos necesiten asistencia; podemos recordarles que el análisis de las características fijas y variables de los escenarios que han hecho en los desafíos resueltos anteriormente.



Solución posible con bloque si, si no al desafío ***Las estrellas de Mañic***.



Solución posible sin bloque si, si no al desafío ***Las estrellas de Mañic***.

Cuando hayan terminado de programar la solución y la hayan puesto a prueba, recuperamos la importancia del trabajo previo a la programación:

- conocer todas las variaciones posibles de un problema e identificar qué aspectos varían y cuáles permanecen fijos,
- pensar una estrategia antes de comenzar a programar y
- programar la estrategia mediante el uso de procedimientos y repeticiones con nombres descriptivos que mejoren la legibilidad del programa.

Promovemos un intercambio en el que cada grupo presente su estrategia y el modo en el que implementaron los bloques de alternativa condicional y los sensores.

¿Qué bloque/s usaron para que Mañic pueda avanzar por uno u otro camino? ¿Utilizaron todos los bloques disponibles en el desafío? ¿Qué ventajas y desventajas ven en utilizar cada uno?

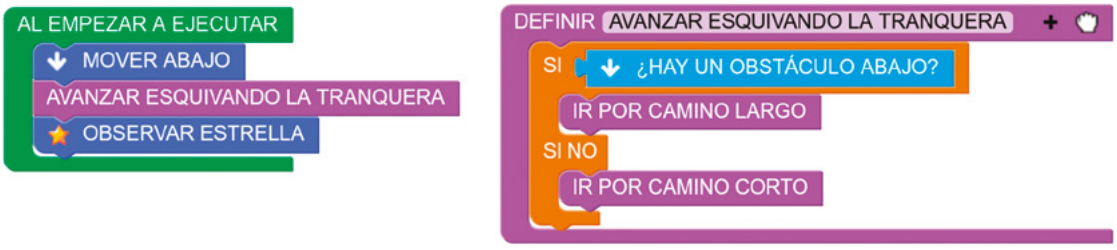
Podemos encontrarnos con diferentes estrategias de solución:

- Los grupos que utilizaron sólo el bloque si habrán colocado dos bloques de este tipo con dos condiciones con sensores.
- Los grupos que hayan usado el bloque si, si no habrán podido resolverlo con un único bloque que considerara las dos ubicaciones posibles de la tranquera.

Utilizar el bloque si, si no permite lograr un programa más compacto. Expresa la idea de que para resolver el problema se debe evaluar una condición para determinar cuál acción ejecutar (por ejemplo, emprender el camino corto o el camino largo); siempre se ejecuta una acción de entre dos posibles. En cambio, el bloque si expresa que una acción se ejecuta dependiendo de si se cumple una condición (y no se ejecuta nada si no se cumple).

¿Utilizaron procedimientos? ¿Para qué? ¿De qué manera?

También podemos comparar las soluciones en función del uso de procedimientos. Por ejemplo, teniendo en cuenta la división en subproblemas, podemos definir un procedimiento para recorrer cada camino. Es probable que haya quienes también definieron un procedimiento que contenga una alternativa condicional para separar el subproblema de "Avanzar esquivando la tranquera" independientemente de qué camino se encuentre habilitado. Esta es una solución más refinada y a la que apunta el desafío siguiente. No es determinante que todos los grupos hayan llegado a esta conclusión en este momento.



Fragmento de la solución con un procedimiento que resuelve la elección y el avance por el camino habilitado.

Brindamos un momento para que los grupos que lo consideren necesario incorporen mejoras a sus programas de acuerdo con las experiencias compartidas. Cuando consideren que el programa está terminado, les pedimos que guarden la solución del desafío para utilizarla en el próximo desafío. Sugerimos a los grupos que **presten atención a la ubicación y el nombre del archivo para que puedan recuperarlo fácilmente**.

En la barra superior del entorno, se encuentra el botón para guardar el desafío. Al cliquearlo, se descarga un archivo con extensión .spbq ('solución de Pilas Bloques').

A continuación, proponemos a los grupos resolver el desafío de Pilas Bloques ***La estrella especial***. Habilitamos un momento de exploración del escenario, enfatizando el uso del botón Ejecutar para conocer los diferentes escenarios y analizar la forma del recorrido.

Escenarios posibles del desafío ***La estrella especial***. La tranquera se ubica en diferentes lugares y la estrella se mantiene en la misma posición.

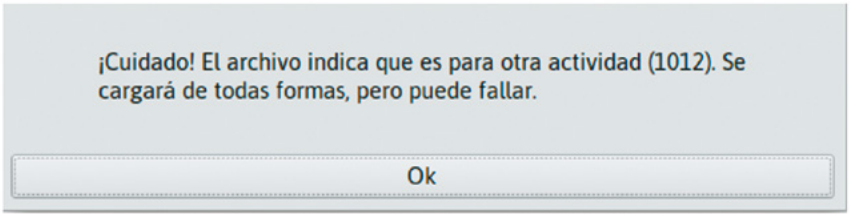
¿Encuentran alguna similitud con el desafío que resolvieron anteriormente?

Apuntamos a reforzar la importancia del análisis del problema para desarrollar **una estrategia de solución antes de programar**. Con las preguntas buscamos que las y los estudiantes reconozcan que el nuevo desafío es similar al anterior: el nuevo sendero está conformado por dos tramos idénticos al sendero del desafío anterior y los obstáculos y la estrella se encuentran en posiciones similares.

¿Algo del programa que construyeron para resolver el desafío anterior (**Las estrellas de Mañic**) *se puede aprovechar para resolver este desafío?*

Reconocidas las similitudes, se propone avanzar con la estrategia de solución reutilizando parte del programa del desafío anterior (***Las estrellas de Mañic***). Para eso, primero, deben recuperar el archivo con la solución.

En la barra superior del entorno, se debe cliquear el botón "Abrir archivo " y buscar el archivo con extensión .spbq que se había guardado al terminar el desafío *Las estrellas de Mañic*.

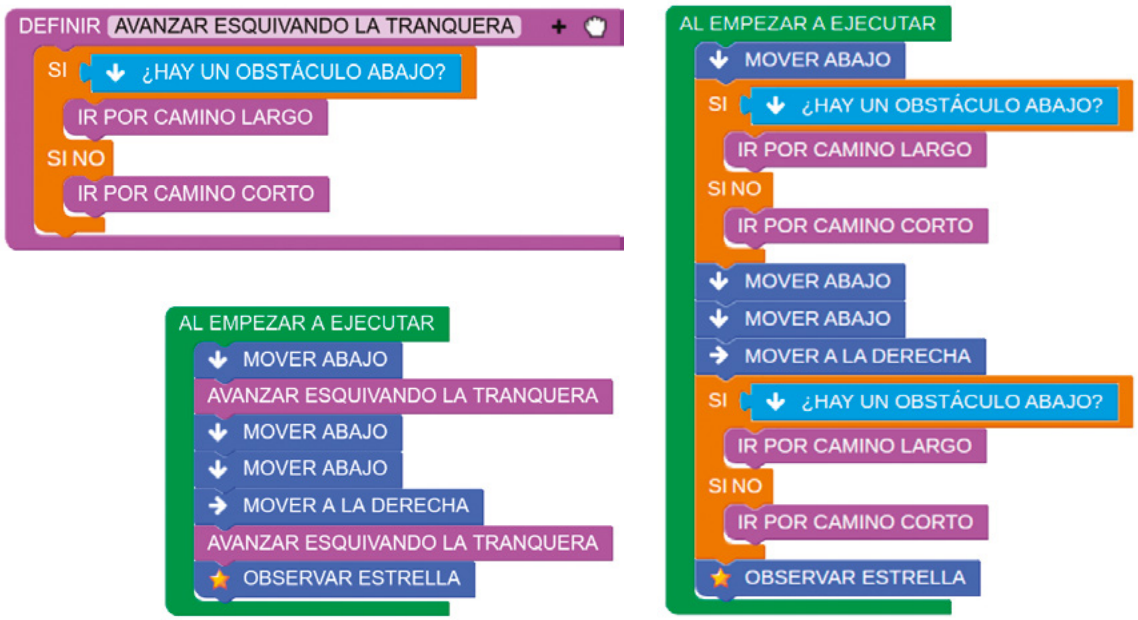


Como el archivo a importar no corresponde al desafío actual, aparecerá un mensaje de advertencia del entorno. Se debe cliquear Ok y continuar.

Cuando los desafíos tienen las mismas primitivas, se puede importar y utilizar un programa ya construido.

Mientras los grupos avanzan en la elaboración de la solución, podemos acompañarlos indicando lo siguiente:

- Pueden utilizar los procedimientos Ir por camino corto o Ir por camino largo (o los nombres que les hayan dado). Si no hubieran definido procedimientos para estas acciones, es una buena oportunidad para definirlos dado que el programa para este desafío será más complejo y, por lo tanto, debemos reforzar su legibilidad.
- Dado que van a utilizar un mismo fragmento de programa dos veces, es posible definir un procedimiento que represente este subproblema y contenga la decisión y el avance por el camino correspondiente. Podemos retomar las discusiones sobre las soluciones al desafío anterior para valorar la definición de ese procedimiento en términos de reutilización.



Soluciones posibles (fragmento) al desafío que aprovechan el programa del desafío anterior.

Cierre >

El **propósito de este momento** es enmarcar la experiencia de la actividad en la idea de la reutilización de soluciones a subproblemas para valorar su importancia y utilidad en el proceso de programación.

Orientaciones

En el desafío **Las estrellas de Mañic**, *¿qué observaron y qué descubrieron que les permitió construir un único programa para resolver el desafío?*

De esa experiencia, nos interesa recuperar la idea de que la alternativa condicional permite resolver situaciones problemáticas variables con un único programa. Para esto, es fundamental analizar los escenarios posibles para identificar qué elementos permanecen constantes, cuáles partes varían y de qué dependen estas variaciones. También hicieron esto en el segundo desafío (***La estrella especial***), junto con otras complejidades.

¿Quiénes definieron un procedimiento para avanzar esquivando la tranquera en el primer desafío? ¿Quiénes lo hicieron cuando empezaron a adaptar la solución del primer desafío al segundo (**La estrella especial**)*? ¿Quiénes al final? ¿Por qué lo hicieron en ese momento? ¿Qué dificultades y ventajas les trajo hacerlo de esa manera?*

Invitamos a los grupos a que cuenten cuándo definieron el procedimiento para avanzar esquivando la tranquera, para señalar **la importancia de la definición de procedimientos en la reutilización**. Quienes ya lo tenían

definido, pueden relatar que construir el programa para el segundo desafío fue particularmente sencillo; quienes lo definieron en este proceso, pueden exponer las razones por las que lo hicieron y los indicios que encontraron para hacerlo, para reforzar la noción de reutilización y su relación con la definición de procedimientos; quienes lo definieron al final pueden rescatar **la importancia de la legibilidad y la posibilidad de refinar los programas ya construidos**[2].

[2] Podemos recuperar las experiencias sobre la reutilización de programas de lo abordado en la secuencia "Programamos en papel cuadriculado. Legibilidad y reutilización" de esta colección, que se encuentra disponible en el sitio ***curriculum.program.ar***.

Actividad 3
Hilera de latas + Turistas latosos

A partir de la solución de dos desafíos de Pilas Bloques (***Hileras de latas*** y ***Turistas latosos***), se retoma el trabajo de creación de estrategias de solución usando alternativas condicionales. Se pone la lupa en la observación de los escenarios para reconocer características variables de un problema para diseñar una estrategia de solución que las atienda y que puedan construirse reutilizando soluciones ya programadas.

Objetivos >

Se espera que las y los estudiantes:

- Refuercen el diseño de estrategias de solución para problemas variables.
- Diseñen e implementen estrategias de solución reutilizando programas.

Inicio >

El **propósito de este momento** es explorar un desafío para relevar las variables que haya que considerar para el diseño de estrategias de solución adecuadas.

Orientaciones

Invitamos a los grupos a ingresar al desafío de Pilas Bloques ***Hilera de latas*** y observar todos los escenarios posibles cliqueando el botón Ejecutar. En esta etapa de exploración, una vez más, buscamos que los grupos reconozcan qué se conserva y qué varía en los diferentes escenarios.

Escenarios del desafío ***Hilera de latas***. La disposición de los casilleros es la misma. La diferencia es que puede haber o no una columna central de latas.

Para aquellos que necesiten ayuda, los acompañamos en el análisis de los dos escenarios, fundamentalmente, para identificar de qué manera podemos darnos cuenta si existe o no una columna de latas en el camino central. Como la columna está completa o no existe en modo alguno, si tenemos una lata en el segundo casillero a la izquierda de la posición inicial de los personajes, es porque existe la columna de latas. También podemos identificar que la lata suelta del extremo izquierdo siempre está.

Retomando todo lo aprendido, es conveniente que alentemos a esbozar una estrategia de solución que, además de servir para todos los casos posibles, esté implementada con procedimientos que puedan mejorar la legibilidad del programa que construirán.

Desarrollo >

El **propósito de este momento** es que las y los estudiantes adquieran autonomía en la solución de los desafíos y que recuperen lo trabajado en las actividades anteriores a propósito de la división en subproblemas con alternativa condicional y la reutilización de soluciones en la construcción de los programas.

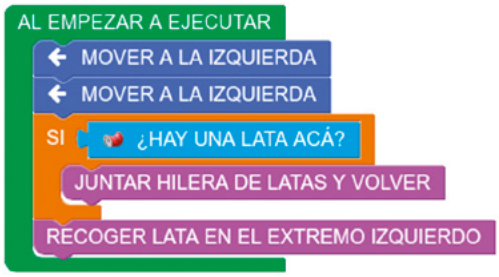
Orientaciones

Mientras los grupos resuelven el desafío, recorremos los puestos de trabajo para asistir en los casos que sea necesario.

Cuando hayan resuelto el desafío, podemos invitar a los grupos a compartir sus programas y repasar el uso de los bloques de alternativa y fomentar que compartan cómo los han utilizado.

¿Qué solución les parece más adecuada? ¿Por qué?

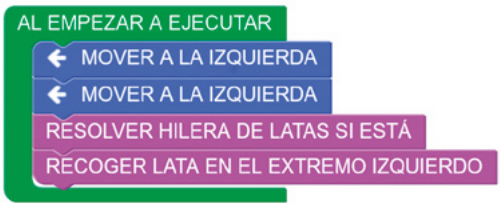
Sobre el uso del bloque si, podemos mencionar que nos permite concentrar la solución al subproblema de la hilera de latas independientemente del de Recoger lata en el extremo izquierdo (en la solución con si, si no debemos repetir el procedimiento Recoger lata... en ambos casos). También, podemos observar que para resolver este subproblema no es necesario elegir entre dos opciones de solución (como en el caso del camino largo y el camino corto de los desafíos de la **Actividad 2**), sino que es necesario realizar una tarea (recoger la hilera de latas) en algunas ocasiones y, en otras, no hay que hacer nada.



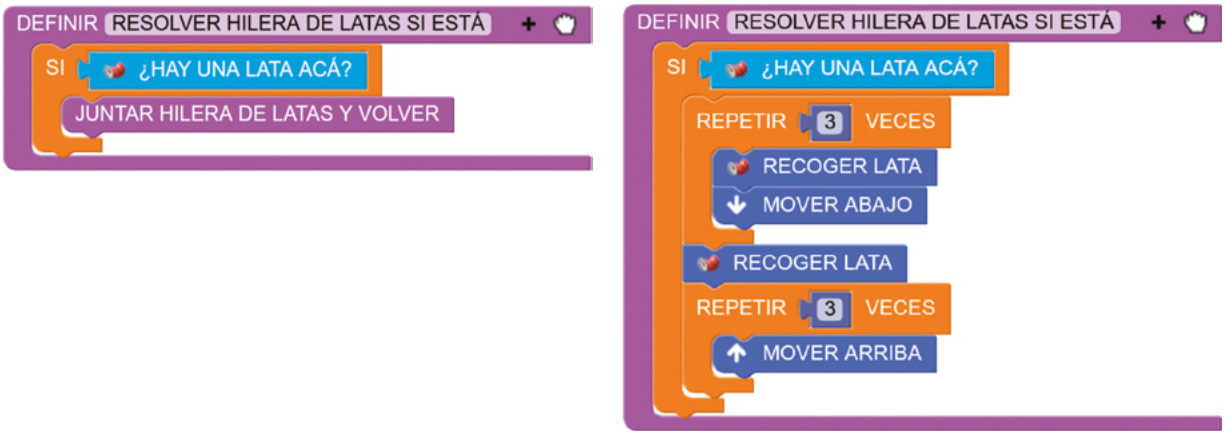
Dos soluciones (fragmento) al desafío ***Hileras de latas*** que utilizan los distintos bloques de alternativa condicional.

Definición de los procedimientos utilizados en ambas soluciones.
Dos soluciones al desafío ***Hilera de latas*** que utilizan distintos bloques de alternativa condicional.

Enfatizando la importancia de descomponer el desafío en subproblemas, podemos alentar la definición de un procedimiento para resolver el subproblema (variable) de la hilera de latas, al igual que lo hicimos en la actividad anterior.



Solución (fragmento) de la solución al desafío con un procedimiento que captura la solución al subproblema variable de recoger las latas cuando aparecen.



Dos definiciones posibles (con distintos niveles de anidamiento) del procedimiento que resuelve la hilera de latas (esté o no presente). Vemos cómo el bloque si permite expresar que solo en determinados casos hay que realizar la tarea. En el primer caso, aprovechamos la definición del procedimiento juntar hilera de latas elaborada previamente.

Estos refinamientos de la solución son deseables, pero no esperamos que todos los grupos los logren autónomamente en esta instancia, que es inicial. Si bien es importante que surjan en la discusión, no hace falta detenernos para que mejoren sus programas en ese momento. Se buscará que logren estos refinamientos en el desafío siguiente (***Turistas latosos***).

Antes de continuar con el próximo desafío, nos aseguramos de que guarden la solución (y presten atención al nombre y ubicación del archivo), ya que podrán reutilizarla en otros desafíos.

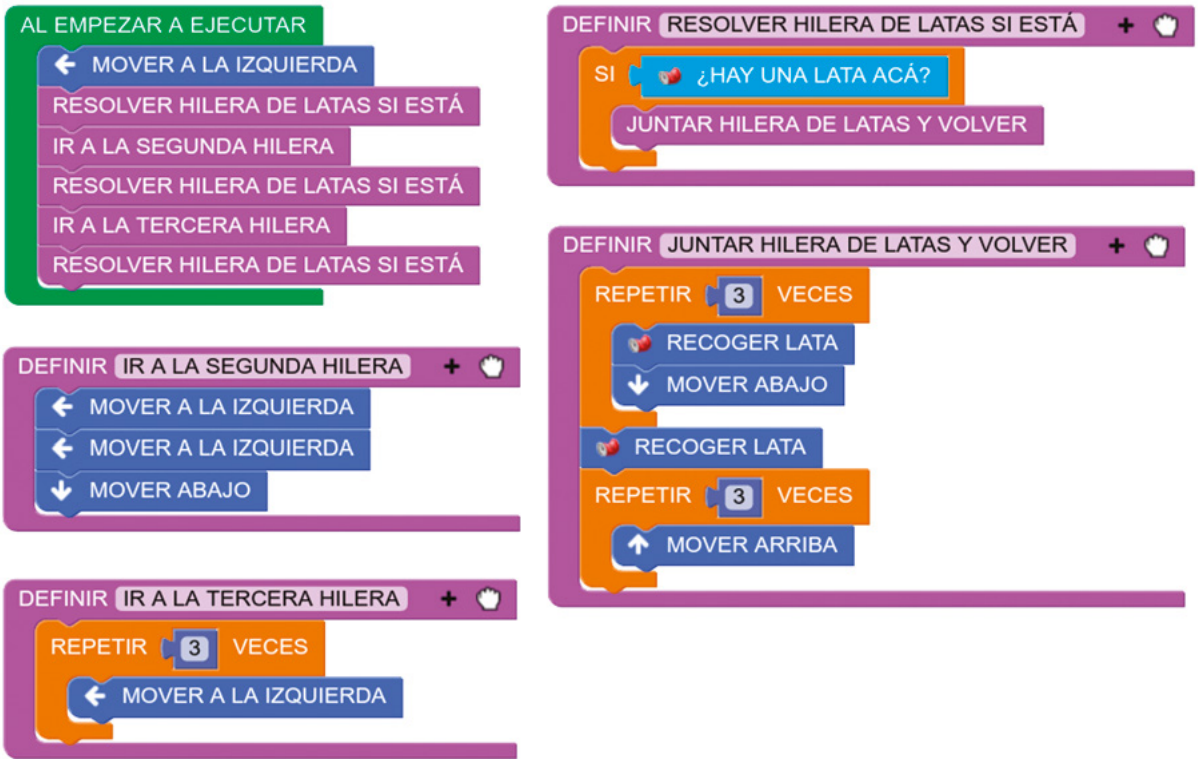
Invitamos a los grupos a explorar autónomamente el desafío ***Turistas latosos***.

Algunos de los escenarios posibles del desafío ***Turistas latosos***.

Nos interesa que los grupos logren el análisis de todos los escenarios posibles. Para ello, podemos recorrer los grupos y pedirles que nos describan el escenario de manera general (por ejemplo, como "un recorrido con tres hileras en las que a veces aparecen latas") y que nos comenten si encuentran similitudes con otros desafíos.

Cuando los grupos hayan arribado a un boceto de la estrategia de solución, podemos proponerles que importen el programa de otro desafío que crean que pueda servirles para resolver este.

Motivamos a que suban la solución (el archivo con extensión .spbq) y definan procedimientos para reutilizar los fragmentos de esta que consideren pertinentes.



Solución completa para el desafío ***Turistas latosos***. Los procedimientos Resolver hilera de latas si está y Juntar hilera de latas y volver están reutilizados de la solución al desafío ***Hilera de latas***.

Cierre >

El **propósito de este momento** es que las y los estudiantes comparen los diferentes usos de los bloques de alternativa condicional y habilitar un momento de metacognición para que identifiquen en su experiencia de programación los conceptos abordados.

Orientaciones

Invitamos a los grupos a que compartan sus soluciones y relaten sus experiencias con los desafíos de Pilas Bloques ***Hilera de latas*** y ***Turistas latosos***, especialmente en cuanto al diseño de la estrategia de solución, la identificación de las características fijas y variables, las decisiones para elegir entre los bloques si y si, si no y el proceso de reutilización de soluciones.

Sobre la elección del bloque si, retomamos la idea de que es útil cuando un subproblema a veces está presente y a veces no ("**Si** hay lata, recogemos la hilera", pero **si no** hay lata simplemente continúa la ejecución del programa). Podemos contrastarlo con el uso del bloque si, si no de la **Actividad 2** (***Las estrellas de Mañic*** y ***La estrella especial***), en el que vieron casos en los que siempre había que hacer algo, pero había que elegir entre dos alternativas (ir por el camino corto o ir por el camino largo). Este espacio de comentar y comparar resultados puede ampliarse con algunas preguntas.

¿Podríamos haberlo resuelto solamente con bloques si o solo con bloques si , si no? ¿Qué cambiaría? ¿En qué afectaría a la estrategia?

Como conclusión de secuencia nos interesa reflexionar sobre los **problemas variables**:

- Hay herramientas de programación que nos permiten resolver problemas variables, es decir, construir **una única solución que funciona para una variedad de casos**. Podemos recuperar las nociones de sensores y alternativa condicional puestas en práctica en los desafíos de Pilas Bloques.
- En la vida contemporánea, encontramos **situaciones en las que un mismo programa resuelve un problema computacional variable**. Alentamos a las y los estudiantes a que identifiquen situaciones en su vida. Podemos citar como ejemplo el programa que controla el dispositivo de una tarjeta de pago de boleto de transporte o similar: el dispositivo lee la información y realiza una acción en función de una condición (si el saldo de la tarjeta es superior o igual al valor del boleto, descuenta su valor; si el saldo es inferior al valor del boleto, arroja

un mensaje de advertencia por saldo insuficiente). Podemos analizar este caso e identificar con las y los estudiantes cómo podría usarse el uso del bloque si, si no para resolver este problema (si hay saldo, cobrar el boleto; si no informar error).

- Siempre que recurrimos a una alternativa condicional es porque hay **determinadas características del problema que permanecen fijas y otras que son variables** y que diferenciarlas es un paso fundamental para resolver el problema. Para reforzar esto, podemos recuperar el análisis de los escenarios de los desafíos y analizar las situaciones de la vida cotidiana que se hayan mencionado (por ejemplo, en el caso del transporte, la persona que opera el dispositivo selecciona la tarifa para cada destino y quien viaja acerca la tarjeta al lector, pero varía el resultado: cobrar la tarifa y habilitar el paso o informar el mensaje de advertencia por saldo insuficiente).

Secuencia didáctica 9

Resolvemos recorridos cambiantes

Alternativa condicional y repetición

¿Podemos resolver un desafío cuyo recorrido cambia en cada ejecución? ¿Y uno con objetos que aparecen y desaparecen en cada ejecución?

En esta secuencia, las y los estudiantes resuelven desafíos en los que las y los personajes deben recorrer trayectos que varían. Para lograrlo con un único programa, deberán combinar alternativas condicionales con repeticiones.

Actividad 1

A partir de los desafíos de Pilas Bloques ***Jugadore de toda la cancha*** y ***Barrilete cósmico***, las y los estudiantes utilizan la alternativa condicional combinada con el bloque de repetición para resolver un recorrido cuyo trazado varía en cada ejecución.

Actividad 2

Para resolver los desafíos de Pilas Bloques ***Alineando telescopios*** y ***Tomando buenas fotos***, las y los estudiantes elaboran una solución que combina repeticiones y alternativas para recorridos que se mantienen constantes pero que requieren acciones variables en cada paso.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: procedimientos, repeticiones, alternativa condicional, sensores.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad.

Objetivos de aprendizaje

- Identificar características relevantes, subproblemas repetidos y características fijas y variables en los escenarios de los desafíos para elaborar una estrategia de solución e implementarla definiendo procedimientos con denominaciones representativas.
- Crear programas en entornos de enseñanza de programación por bloques que combinen comandos primitivos, alternativas condicionales, repeticiones simples y sensores para resolver problemas con escenarios variables.
- Caracterizar los escenarios de un desafío como una repetición fija de casos variables y proponer una estrategia para resolverlo que utilice la alternativa condicional como parte de una repetición.

Saberes previos de CC

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, procedimientos, repeticiones, alternativa condicional, sensores.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad.

Materiales necesarios

- Dispositivos con Pilas Bloques instalado o acceso a su versión en línea https://pilasbloques.program.ar/

Actividad 1
Jugadore de toda la cancha + Barrilete cósmico

Las y los estudiantes se aproximan al problema de los recorridos con escenarios variables con el desafío ***Jugadore de toda la cancha***, en el que deberán conseguir que Chuy avance un casillero en una dirección que cambia en cada ejecución. Luego, para resolver el desafío ***Barrilete cósmico***, reutilizan la solución del primer desafío y la adecuan para completar un recorrido más largo.

Objetivos >

Se espera que las y los estudiantes resuelvan problemas de recorridos cambiantes de longitud fija y comiencen a delinear una estrategia general para recorridos cambiantes que tenga como subtarea una toma de decisión con respecto al avance.

Inicio >

El **propósito de este momento** es explorar el primer desafío para reconocer que los escenarios varían y esbozar una solución que contemple las variaciones.

Orientaciones

Organizamos la clase en grupos heterogéneos pequeños[1]. Invitamos a que abran el desafío ***Jugadore de toda la cancha.***

¿Cuántos escenarios posibles hay? ¿Con qué sensores y primitivas cuentan?

A partir de estas preguntas, buscamos que identifiquen los dos escenarios posibles del problema: avanzar hacia la derecha o hacia abajo. A continuación, les pedimos que esbocen una posible estrategia de solución.

[1] Se pueden consultar dinámicas lúdicas para el armado de grupos heterogéneos en "Enseñar computación desde la mirada de la Educación Sexual Integral (ESI)", 10.

Los dos escenarios posibles del desafío ***Jugadore de toda la cancha***.

Desarrollo >

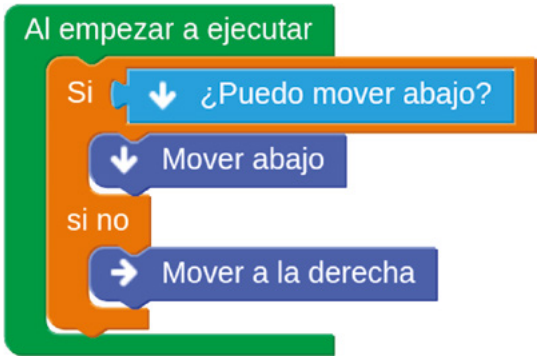
El **propósito de este momento** es aproximarnos a una solución para recorridos cambiantes usando las primitivas y los sensores disponibles, reforzando la necesidad de diseñar soluciones que contemplen todos los escenarios posibles y valorando la elaboración de programas que puedan ser reutilizados.

Orientaciones

Invitamos a los grupos a que avancen en la solución del desafío ***Jugadore de toda la cancha*** en Pilas Bloques.

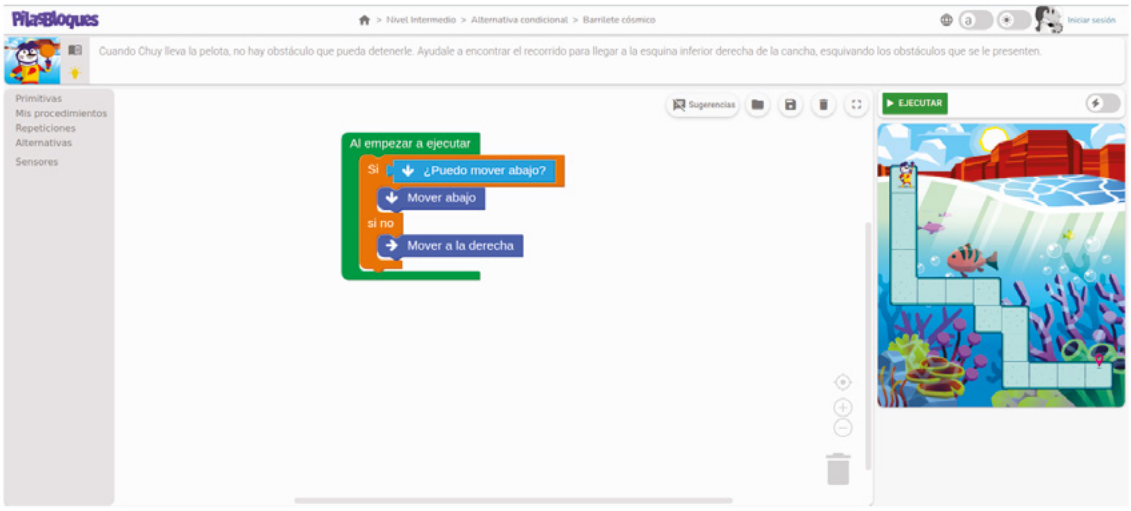
Mientras recorremos los puestos de trabajo, vamos charlando con las y los estudiantes y observamos la estrategia de solución propuesta con atención en el uso del bloque de alternativa condicional y los sensores disponibles. En caso de advertir que no están usando el bloque Si, si no, podemos evocar algún desafío previo en el que lo hayan utilizado para atender dos alternativas, por ejemplo, ***¿Pelota o paleta?***

Una vez que hayan resuelto el desafío, y antes de pasar al siguiente momento, les pedimos que guarden la solución en una ubicación que luego puedan hallar. Será un insumo para el próximo desafío.



Una solución posible al desafío ***Jugadore de toda la cancha***.

A continuación les proponemos a los grupos que ingresen al desafío ***Barrilete cósmico*** y abran la solución guardada del desafío anterior. El objetivo es pensar la manera de aprovechar ese programa para resolver el nuevo desafío.

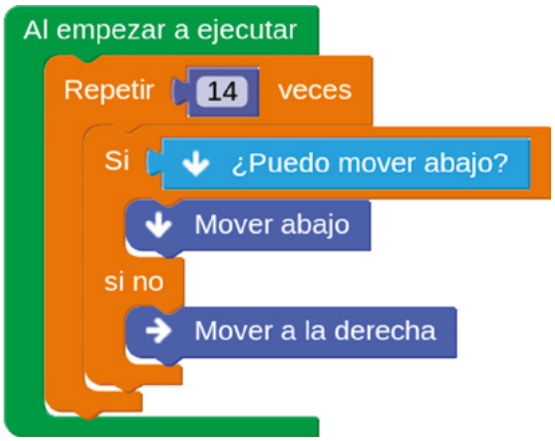


El desafío ***Barrilete cósmico*** con la solución al desafío anterior ya importada.

Promovemos que el primer paso sea la exploración del desafío para que identifiquen que el laberinto, por más que cambie su recorrido, siempre se compone de 14 casilleros. Otro aspecto fundamental que deben identificar es que el próximo casillero del recorrido siempre aparece debajo o a la derecha, al igual que el segundo casillero del desafío que resolvieron anteriormente. Esta es la clave que les permitirá reutilizar su solución. Podemos motivar esta instancia de análisis del problema con las siguientes preguntas.

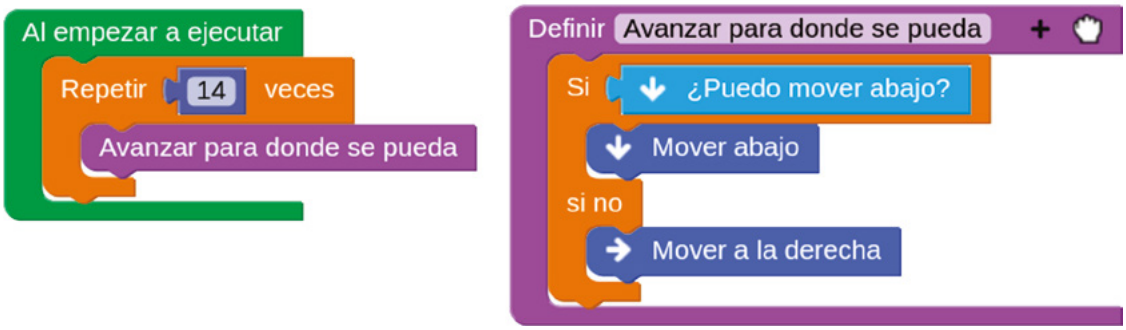
Ejecuten varias veces el desafío. ¿Observan cambios en el recorrido? ¿Qué se mantiene y qué varía? ¿Cómo describirían el recorrido? ¿En qué se parece al recorrido del desafío anterior y en qué es diferente? ¿Qué pueden aprovechar de lo que programaron antes?

El objetivo de este desafío es que reconozcan que el escenario nuevo es una repetición (de longitud constante) del escenario anterior, para motivar la combinación de la alternativa condicional con la repetición para resolver problemas variables. En este caso particular, es posible construir la nueva solución agregando un bloque de repetición.



Una solución posible al desafío ***Barrilete cósmico***.

Recorremos los grupos y conversamos con ellos. Es posible que algunos lleguen a un refinamiento de la solución usando la definición de un procedimiento para capturar la acción de avanzar en la dirección habilitada. Para que todos lo logren, podemos recuperar desafíos anteriores en los que definieron procedimientos para resolver una tarea pequeña de una manera particular (por ejemplo, Avanzar en diagonal en el desafío ***Yvoty y las luciérnagas***).



Solución al desafío ***Barrilete cósmico*** que contempla la definición de un procedimiento para la tarea de avanzar un casillero.

Cierre >

El **propósito de este momento** es explicitar que la estrategia utilizada en el desafío ***Barrilete cósmico*** consistió en la repetición de la solución a un subproblema variable.

Orientaciones

¿En qué consistió la solución al primer desafío? ¿Y el segundo? ¿Qué parecidos tienen ambas soluciones?

A modo de plenario, retomamos los pasos que se transitaron para construir la solución para el desafío ***Barrilete cósmico***. Se reutilizó la solución de ***Jugadore de toda la cancha***, diseñada para un escenario de dos casilleros: para el segundo desafío, fue necesario repetir esta solución (que podría definirse como un procedimiento para lograr aún mayor legibilidad del programa). Si describimos la estrategia, podemos identificar que consistió en resolver un subproblema varias veces, y que el subproblema consistía en "pasar al siguiente casillero", teniendo en cuenta que el casillero podía aparecer debajo o a la derecha.

La diferencia con otros desafíos con repetición, que resolvimos en secuencias didácticas anteriores, es que el subproblema que se repite en este caso es variable (de acuerdo a la posición del casillero) y, por lo tanto, requiere el uso de una alternativa condicional.

¿Qué situaciones de la vida cotidiana se les ocurren que podrían ser un problema que se resuelva repitiendo una solución con alternativa condicional?

Alentamos la generalización de esta situación en situaciones de la vida de las y los estudiantes. Por ejemplo, al tomar asistencia en el aula, repetimos la acción de leer un nombre de una lista y anotar "Presente" o "Ausente" según si vemos que está presente o no.

Actividad 2
Alineando telescopios + Tomando buenas fotos

Los y las estudiantes resuelven dos desafíos que retoman la idea de recorrido, pero introduciendo la novedad de que los casilleros pueden tener un objeto o no. Esto requiere estrategias que, en cada paso, realicen una evaluación antes de ejecutar una acción.

Objetivos >

Se espera que las y los estudiantes:

- Amplíen el reconocimiento de características fijas y variables entre diferentes escenarios posibles.
- Combinen repeticiones y alternativas condicionales en la elaboración de soluciones.
- Extiendan estrategias de solución conocidas que incluyen la realización de una acción en todos los pasos de un recorrido a desafíos en los que esta acción es necesaria sólo en algunos casilleros.

Inicio >

El **propósito de este momento** es que los y las estudiantes exploren los desafíos.

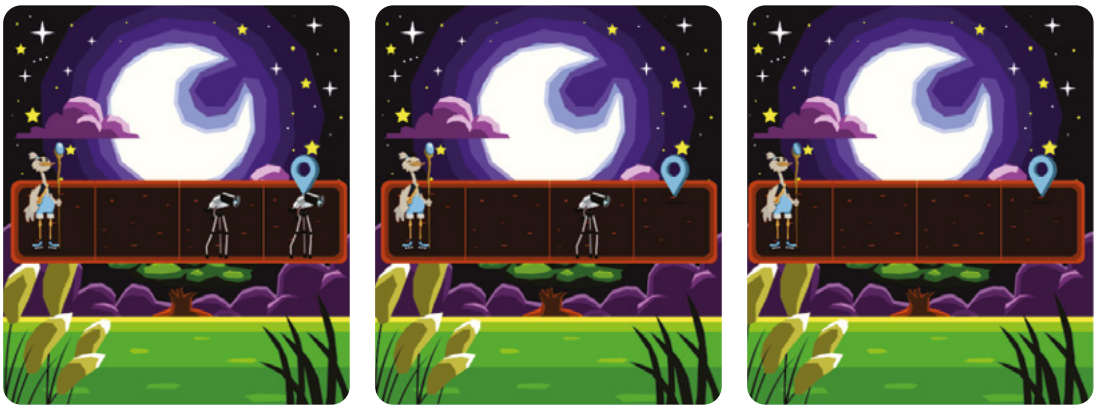
Orientaciones

Presentamos a los grupos los desafíos ***Alineando telescopios*** y ***Tomando buenas fotos*** y les proponemos que exploren los escenarios y elijan por cuál desafío comenzar. Es importante que reconozcan la novedad: la presencia de los objetos en los casilleros varía en cada ejecución.

¿Qué elementos permanecen constantes en cada ejecución? ¿Cuáles cambian? ¿Qué es lo que cambia?

Se puede orientar la exploración con las preguntas anteriores, para que identifiquen que la forma y el tamaño de los recorridos permanecen constantes y tanto la cantidad y la posición de los objetos varían. Este análisis es fundamental para la elaboración de una estrategia que incluya la repetición de una acción variable.

Escenarios posibles del desafío ***Tomando buenas fotos***.



Escenarios posibles del desafío ***Alineando telescopios***.

Desarrollo >

El **propósito de este momento** es reforzar la identificación de regularidades y variaciones en un escenario para resolver desafíos de recorridos cambiantes. También, brindar un espacio para socializar las experiencias, ya que no todos los grupos habrán resuelto los desafíos en el mismo orden.

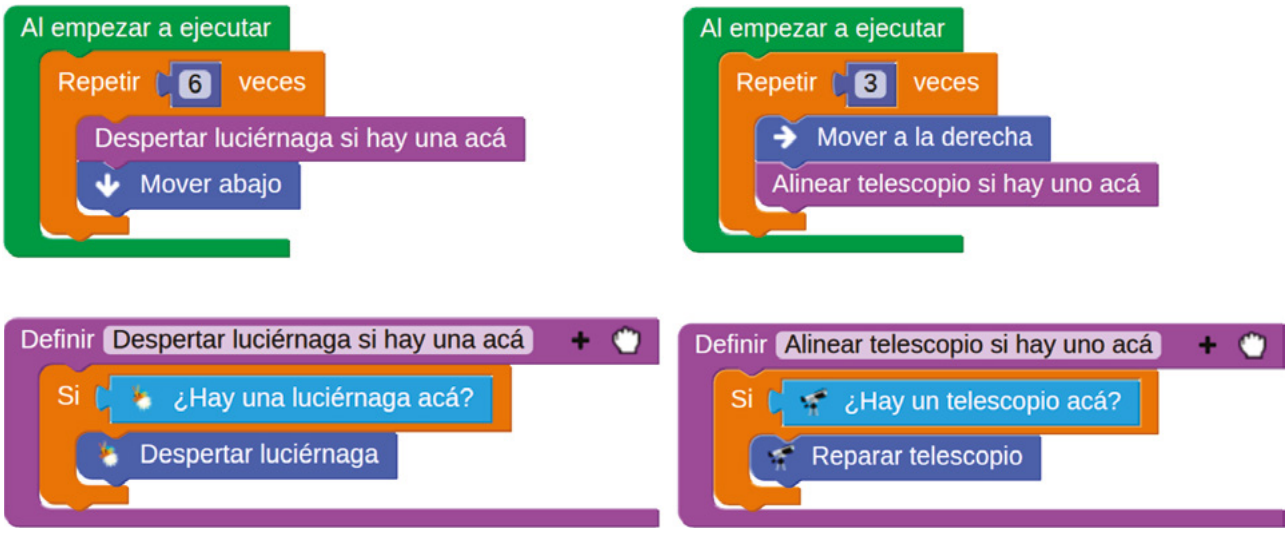
Orientaciones

Luego de que cada grupo haya elegido por cuál desafío comenzar y hayan bocetado las soluciones, recorremos los puestos de trabajo prestando especial atención a la elaboración de la estrategia de solución, la combinación de la repetición con la alternativa condicional y la definición de procedimientos.

Cuando los grupos completen la solución de su primer desafío, los invitamos a avanzar con el siguiente. Alentamos a que analicen el escenario e identifiquen parecidos con el que ya resolvieron para tomar ideas para la construcción de la nueva solución.

Podemos brindar un espacio común para evacuar dudas, pero teniendo mucho cuidado de no adelantar soluciones o ideas clave a los grupos que aún no resolvieron alguno de los desafíos. Esto será tema de la puesta en común de cierre.

Una observación para tener en cuenta es que, en ***Alineando telescopios***, el casillero inicial nunca está ocupado por un telescopio, pero el último puede estarlo. En cambio, en el desafío ***Tomando buenas fotos***, puede haber una luciérnaga en el casillero inicial, pero nunca aparece una luciérnaga en el último. Es probable que los grupos descubran este detalle cuando el entorno les informe un error en alguna ejecución. Esta dificultad invita a que revisen el orden de ejecución de la acción sobre el objeto y la acción de avanzar dentro de la repetición: si el primer casillero puede estar ocupado, primero debe atenderse la acción sobre el objeto y luego avanzar; si el primer casillero siempre estará libre, se debe avanzar primero y luego atender la acción.



Soluciones posibles a los desafíos ***Tomando buenas fotos*** (izquierda) y ***Alineando telescopios*** (derecha).

Ambas soluciones recurren a la división en subtareas: la repetición de la acción de avanzar y la acción sobre cada casillero (que requiere del uso de la alternativa condicional para decidir si es necesaria o no según haya un objeto presente). Además, esta división está explícita en el programa mediante la definición de procedimientos.

Como indicador de avance podemos observar si las y los estudiantes incorporan estas ideas para construir sus soluciones y las explicitan cuando tienen que comunicar o justificar sus programas.

Cierre >

El **propósito de este momento** es generalizar las estrategias elaboradas a partir de socializarlas y compararlas.

Orientaciones

Los grupos comparten sus soluciones y charlamos sobre los resultados y las dificultades encontradas. Nos interesa que identifiquen similitudes en las estrategias para generalizarlas como un recorrido de longitud fija (que se resuelve con repetición) en el que hay una acción variable en cada casillero (que se resuelve con alternativa condicional), cuyas opciones en este caso son realizar una acción o no hacer nada.

¿Qué tipo de desafíos resolvieron en esta secuencia? ¿Qué parecidos y diferencias encuentran con los que resolvieron en secuencias anteriores? ¿Qué otros desafíos se les ocurre que podrían resolver ahora?

Queremos que las y los estudiantes recuperen sus experiencias durante la secuencia para reconocer dos diferencias clave con secuencias que se trabajaron anteriormente. Por un lado, en la secuencia en la que trabajaron recorridos[2], había un único escenario en el que tanto el tamaño como la forma y el contenido del recorrido era siempre igual; por el contrario, en esta secuencia, si bien el tamaño del recorrido es constante, vieron que la forma o el contenido de los casilleros puede variar. Por otro lado, en la secuencia "¿Cómo se resuelven problemas cambiantes?", se trabaja con escenarios variables que requieren una alternativa condicional en momentos puntuales de la solución para decidir qué camino tomar (en el desafío ***La estrella especial***) o si era necesario recorrer alguna parte adicional del escenario (en el desafío ***Turistas latosos***). En cambio, en esta secuencia, la alternativa condicional se utiliza en todos los pasos del recorrido, pues está asociada a la acción que debe realizarse en todos los casilleros.

Por último, como un paso más de generalización de este esquema, alentamos a las y los estudiantes a que imaginen escenarios de desafíos que podrían resolver con la estrategia de recorridos con casilleros variables. Pueden proponer recorridos similares, pero con otros objetos y personajes y combinar ambas ideas: un recorrido de forma variable en el que aparezcan o no objetos en los casilleros.

---

[2] Abordados en los desafíos ***Ivoyt y las luciérnagas*** o ***Campeone desordenade*** de la secuencia "Estrategias y división en subproblemas" de esta colección, que se encuentra disponible en el sitio curriculum.program.ar.

Secuencia didáctica 10

Programamos estrategias para problemas cambiantes

Estrategia, alternativa condicional y repetición

¿Qué desafíos nuevos podemos resolver combinando las herramientas que ya conocemos? ¿Cómo son las estrategias para esto? ¿Qué procedimientos nos conviene definir?

En esta secuencia, a modo de integración, se abordan desafíos que requieren recuperar y combinar las herramientas de programación abordadas en secuencias didácticas previas de esta colección, con especial atención a la elaboración de estrategias de solución y la definición de procedimientos.

Actividad 1

A partir del desafío de Pilas Bloques ***¿Latas o papeles?***, las y los estudiantes programarán soluciones con una estrategia que priorice la legibilidad y aproveche la reutilización de procedimientos.

Actividad 2

A partir del desafío de Pilas Bloques ***Curvas de celus***, las y los estudiantes refuerzan las ideas de estrategia, división en subproblemas y reutilización para resolver un problema variable con un escenario más complejo.

Actividad 3

A partir del desafío de Pilas Bloques ***Festín astronómico***, las y los estudiantes profundizan el diseño de estrategias que contemplen escenarios más complejos y la aparición de uno u otro objeto en los casilleros.

Actividad 4

A partir del desafío de Pilas Bloques ***Mariposas***, las y los estudiantes refuerzan el diseño de estrategias restringidas a las primitivas disponibles.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: procedimientos, repeticiones, alternativa condicional.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad.

Objetivos de aprendizaje

- Identificar características relevantes, comportamientos repetidos y características fijas y variables en el escenario de los desafíos para elaborar una estrategia de solución a partir de la descomposición del problema en subproblemas que se exprese mediante procedimientos con una denominación representativa (que aporta a la legibilidad del programa).
- Identificar regularidades en un programa y expresarlas a través de las herramientas adecuadas del lenguaje.
- Crear programas en entornos de enseñanza de programación por bloques que combinen procedimientos, repetición simple y alternativa condicional para resolver desafíos.

Saberes previos de CC

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, secuencias, procedimientos, repeticiones, alternativa condicional.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad.

Materiales necesarios

- Dispositivos con Pilas Bloques instalado o acceso a su versión en línea https://pilasbloques.program.ar/

Actividad 1
¿Latas o papeles?

La disposición del escenario del desafío motiva a las y los estudiantes a definir un procedimiento para resolver el mismo subproblema más de una vez. Para hacerlo deberán integrar conceptos abordados en secuencias anteriores, con énfasis en la división de problemas en subproblemas, la legibilidad y la estrategia de solución.

Objetivos >

Se espera que las y los estudiantes:

- Utilicen la técnica de división de un problema en subproblemas para pensar y diseñar estrategias de solución y para construir programas legibles mediante el uso de procedimientos con una denominación representativa y en términos del dominio del problema.
- Identifiquen características relevantes, subproblemas repetidos y características fijas y variables del escenario de un desafío de programación.
- Elaboren una estrategia de solución que atienda escenarios variables.

Inicio >

El **propósito de este momento** es presentar el desafío a resolver y motivar el análisis de los escenarios para identificar regularidades de la grilla y características constantes y variables en vistas a construir una estrategia de solución compacta y legible.

Orientaciones

Organizamos la clase en grupos heterogéneos pequeños[1]. Presentamos el desafío ***¿Latas o papeles?*** y promovemos un momento de exploración de todos los escenarios posibles (cliqueando el botón «Ejecutar») y su análisis con especial énfasis en la identificación de regularidades y características constantes y variables.

[1] Se pueden consultar dinámicas lúdicas para el armado de grupos heterogéneos en "Enseñar computación desde la mirada de la Educación Sexual Integral (ESI)", 10.

Algunos escenarios posibles del desafío ***¿Latas o papeles?***

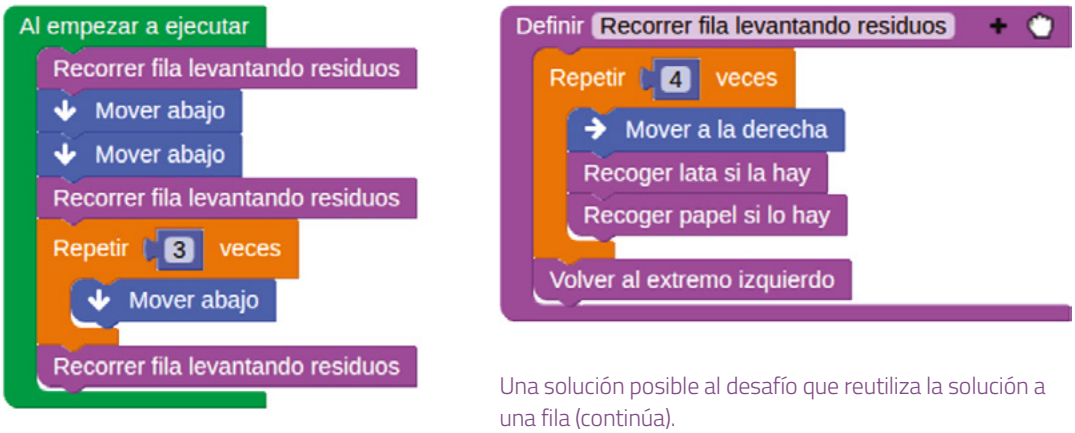
Con la información relevada, les indicamos que reconozcan cómo podría dividirse el problema en subproblemas, y hagan un boceto de una estrategia de solución. A aquellos grupos a los que les haya costado esto, podemos ayudarlos a reconocer que la clave se encuentra en la reutilización del fragmento de programa que resuelva una fila del escenario.

Desarrollo >

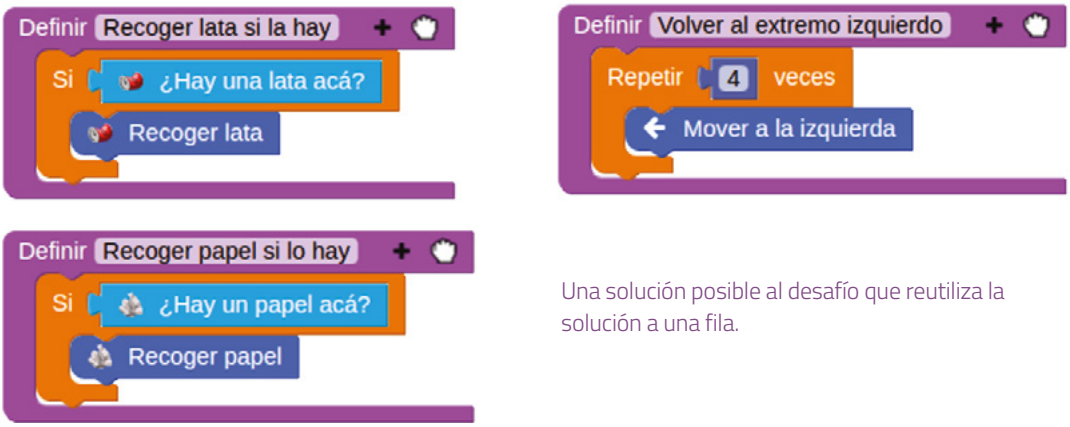
El **propósito de este momento** es brindar una instancia de trabajo para que las y los estudiantes resuelvan autónomamente el desafío, teniendo en cuenta el boceto de estrategia de solución diseñada en el momento anterior.

Orientaciones

Mientras los y las estudiantes trabajan sobre el desafío, recorremos los grupos reforzando la importancia de recuperar las herramientas abordadas en las secuencias anteriores con especial atención a cuestiones de las estructuras de los programas: su legibilidad, la denominación descriptiva de los procedimientos y su reutilización.



Una solución posible al desafío que reutiliza la solución a una fila (continúa).



Una solución posible al desafío que reutiliza la solución a una fila.

Cierre >

El **propósito de este momento** es socializar las estrategias de solución para explicitar las decisiones que favorecieron la reutilización y la legibilidad del programa y, en particular, el uso de la alternativa condicional.

Orientaciones

¿Había partes del recorrido que fueran iguales? ¿Crearon una solución para cada tramo o reutilizaron una solución? ¿Qué tienen en común los tramos? ¿Qué herramienta de programación permitió resolver los tres tramos con un mismo procedimiento?

Las y los estudiantes comparten los resultados y, mediante preguntas orientativas, llevamos el foco del plenario a cuestiones de legibilidad y reutilización. En particular, a cómo el uso de los bloques de alternativa permite identificar como un mismo subproblema las tres filas a pesar de que los objetos en ellas no son los mismos.

Actividad 2
Curvas de celus

Las y los estudiantes ponen en práctica de forma autónoma cómo dividir un problema en subproblemas variables y con la puesta en común refuerzan la importancia de esta tarea al diseñar una estrategia de solución.

Objetivos >

Se espera que las y los estudiantes:

- Utilicen la técnica de división de un problema en subproblemas tanto para pensar y diseñar estrategias de solución como para construir programas que utilicen procedimientos con denominación representativa en términos del dominio del problema.
- Identifiquen características relevantes, subproblemas repetidos y características fijas y variables en el escenario de un desafío de programación.
- Elaboren una estrategia de solución a partir de descomponer el problema en términos de subproblemas.

Inicio >

El **propósito de este momento** es explorar el desafío e identificar las regularidades para diseñar una estrategia de solución.

Orientaciones

Se presenta el desafío ***Curvas de celus*** y, ya en este punto del aprendizaje, proponemos sin más a los y las estudiantes que resuelvan el desafío, pensando una estrategia de solución que considere las regularidades en la disposición de los casilleros y los objetos, prestando atención a características variables y fijas.

Algunos escenarios del desafío ***Curva de celus***.

Desarrollo >

El **propósito de este momento** es brindar un espacio de trabajo autónomo a las y los estudiantes para que resuelvan el desafío y puedan solicitar ayuda.

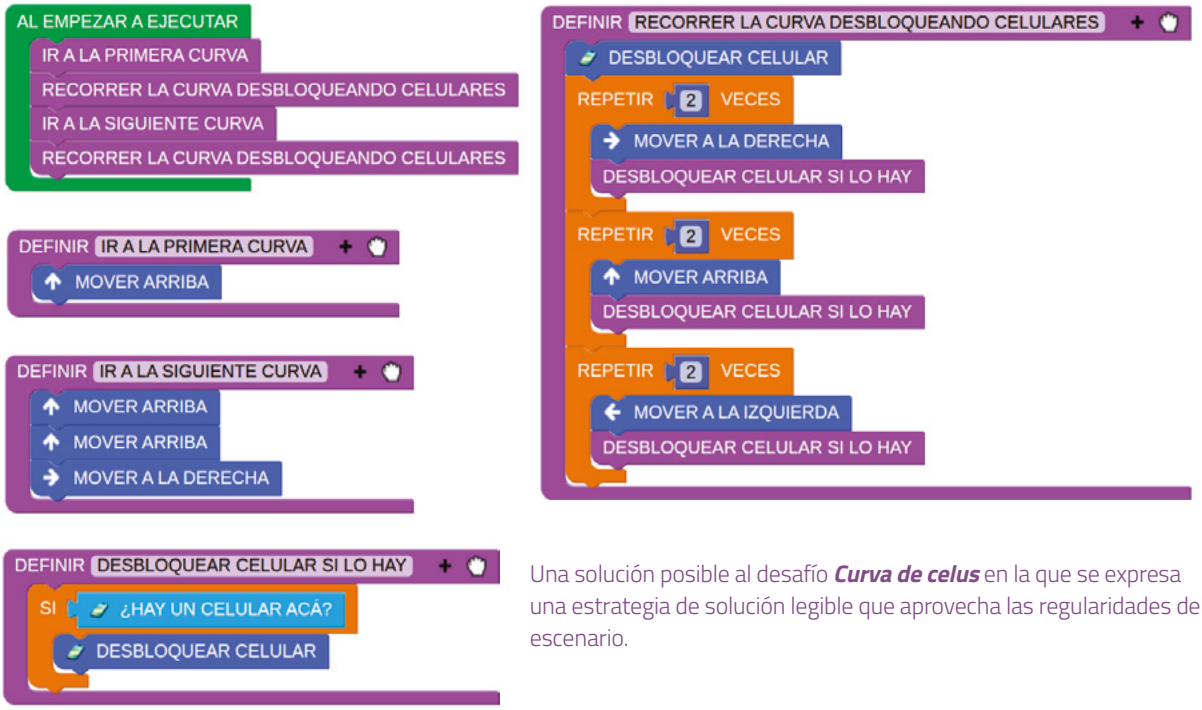
Orientaciones

En esta secuencia didáctica, los desafíos pretenden motivar la recuperación y la integración de las herramientas de programación abordadas en las secuencias anteriores de la colección[2]. El objetivo es que las y los estudiantes puedan hacerlo de manera autónoma y, durante este proceso, es posible que requieran orientación.

Prestando atención a los avances, dificultades y pedidos de ayuda, podemos proponer preguntas para hacer hincapié en la identificación de regularidades y variaciones en los diferentes escenarios y la definición de procedimientos.

¿Hay casilleros en las que siempre hay un celular? ¿Hay casilleros en las que nunca hay un celular? ¿Qué patrones podemos encontrar, respondiendo a las dos preguntas anteriores? ¿Los recovecos del camino tienen puntos o formas en común? ¿Podemos hacer procedimientos que sirvan para determinados tramos y reutilizarlos?

[2] Todas las secuencias de la colección se encuentran disponibles en el sitio curriculum.program.ar.



Una solución posible al desafío ***Curva de celus*** en la que se expresa una estrategia de solución legible que aprovecha las regularidades del escenario.

Cierre >

El **propósito de este momento** es conversar en grupo sobre las estrategias planteadas y las herramientas utilizadas en la implementación para ver ventajas y cuestiones a mejorar en cada una.

Orientaciones

Promovemos el intercambio entre las y los estudiantes para que compartan las diferentes alternativas de solución que diseñaron y programaron. Moderamos el intercambio para poner de manifiesto las decisiones a propósito del uso de los bloques de alternativa Si o Si, si no. También nos interesa recuperar la definición de procedimientos y la división en subproblemas a partir de la observación de que el escenario tiene dos partes muy similares que se repiten y, por lo tanto, pueden ser resueltas con un mismo procedimiento. Es posible que hayan aparecido otros procedimientos para dividir el recorrido de la curva. Alentamos a que se presenten y comenten.

Como cierre podemos repasar los acuerdos en las estrategias propias y de otros grupos que sería recomendable considerar en futuros desafíos.

Actividad 3
Festín astronómico

Al resolver este desafío, las y los estudiantes se enfrentan con un nuevo problema variable que deberán analizar para diseñar estrategias que aprovechen regularidades.

Objetivos >

Se espera que las y los estudiantes:

- Utilicen la técnica de división de un problema en subproblemas tanto para pensar y diseñar estrategias de solución como para crear programas con procedimientos con una denominación representativa en términos del dominio del problema.
- Identifiquen características relevantes, subproblemas repetidos y características fijas y variables en el escenario de un desafío de programación.
- Elaboren una estrategia de solución a partir de descomponer el problema en términos de subproblemas.

Inicio >

El **propósito de este momento** es presentar el desafío y alentar a las y los estudiantes a que autónomamente exploren y analicen el escenario para la definición de la estrategia de solución.

Orientaciones

Las y los estudiantes ingresan al desafío ***Festín astronómico*** y exploran los distintos escenarios. Observamos el trabajo y, si fuera necesario, alentamos con preguntas la identificación de regularidades.

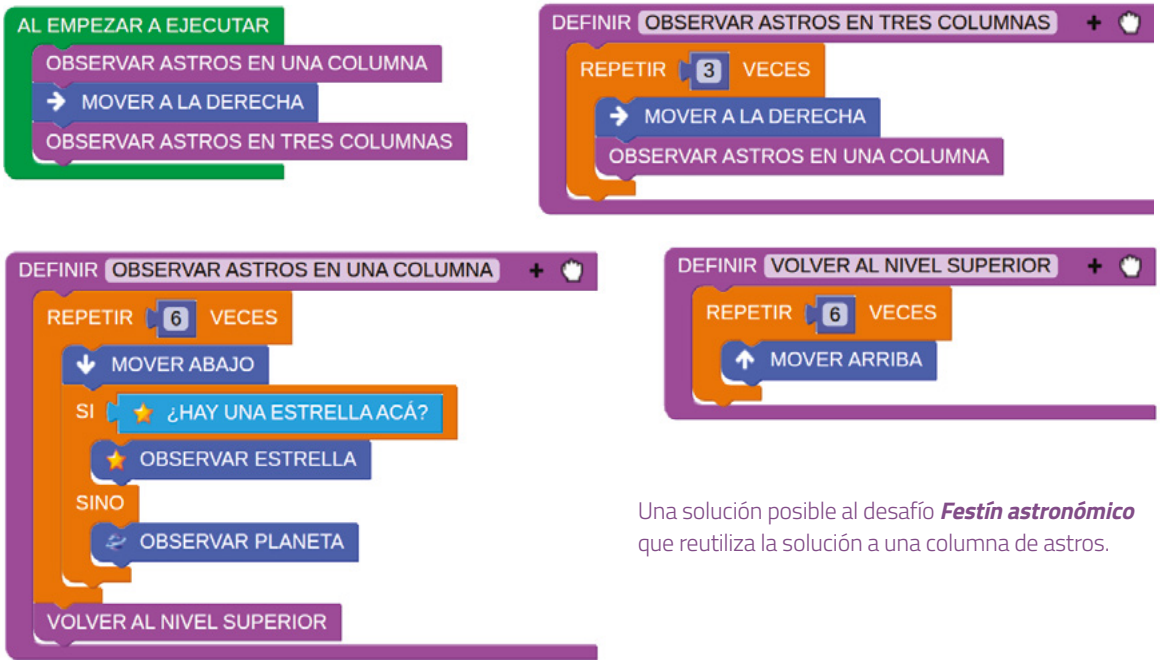
Algunos escenarios posibles del desafío ***Festín astronómico***.

Desarrollo >

El **propósito de momento**, como en toda la secuencia, es que las y los estudiantes recuperen e integren lo aprendido en el resto del trayecto para resolver nuevos desafíos.

Orientaciones

Invitamos a resolver el desafío. Para construir la estrategia, es clave identificar la disposición del escenario como cuatro columnas que se repiten y en las que todos los casilleros están ocupados por un astro. Por lo tanto, es recomendable usar el bloque Si, si no: si no tiene una estrella, tendrá un planeta, pero nunca estará vacía (como sí sucedía en las actividades precedentes de esta secuencia). En caso de ser necesario, podemos orientar a los grupos con preguntas para que arriben a estos puntos.



Una solución posible al desafío ***Festín astronómico*** que reutiliza la solución a una columna de astros.

Cierre >

El **propósito de este momento** es retomar entre todas y todos lo hecho en el desarrollo y socializar las diferentes estrategias para recuperar las ideas claves del desafío: la identificación de regularidades y su aprovechamiento para diseñar la estrategia y crear el programa.

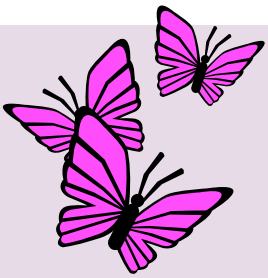
Orientaciones

Invitamos a las y los estudiantes a compartir sus soluciones y sus estrategias. En ellas, repasamos cuestiones como la posibilidad de usar el bloque Si, si no al estar siempre ocupados los casilleros, la reutilización de una parte de la solución al ser todas las columnas iguales (mediante la definición de un procedimiento que resuelve el problema independientemente de si hay planetas o estrellas en los casilleros) y el nivel de legibilidad que puede alcanzarse en el bloque principal de donde puede deducirse muy fácilmente la estrategia utilizada.

Actividad 4
Mariposas

Para cerrar la secuencia, proponemos el desafío ***Mariposas*** en el que las estrategias de solución factibles están limitadas por las primitivas disponibles.

Objetivos >



Se espera que las y los estudiantes:

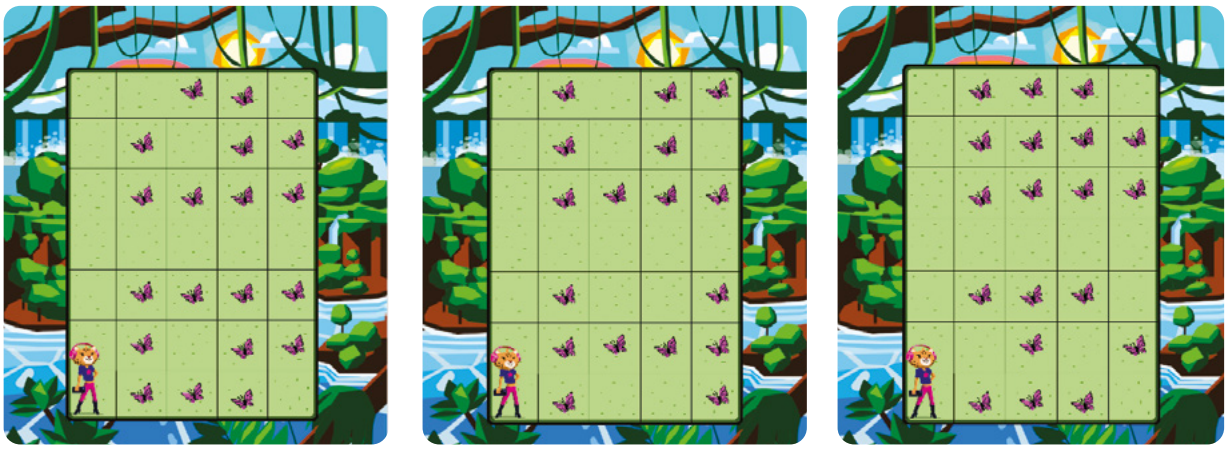
- Elaboren una estrategia de solución para un desafío que pueda ser implementada con las herramientas provistas por el lenguaje de programación.
- Identifiquen características relevantes, subproblemas repetidos y características fijas y variables en el escenario de un desafío de programación.
- Elaboren una estrategia de solución a partir de descomponerlo en términos de subproblemas y considerando las características analizadas.

Inicio >

El **propósito de este momento** es contrastar las estrategias propuestas para resolver un desafío con las primitivas disponibles para resolverlo.

Orientaciones

Mostramos los escenarios del desafío ***Mariposas*** para que las y los estudiantes anticipen estrategias de solución sin conocer las primitivas disponibles.



Algunos escenarios posibles del desafío ***Mariposas***.

Invitamos a que compartan las estrategias pensadas, esperando que surjan estrategias variadas, por ejemplo: recorrer el tablero por filas volviendo al borde izquierdo o en zigzag, también avanzando verticalmente por columnas, etc. Entonces, invitamos a todas y todos a explorar las primitivas disponibles para revisar si la estrategia que habían propuesto es factible.

Primitivas disponibles en el desafío.

Como apoyo, podemos recuperar la experiencia con el desafío ***Reparadora de telescopios*** trabajado en la secuencia didáctica de esta colección "Programamos estrategias en Pilas Bloques".

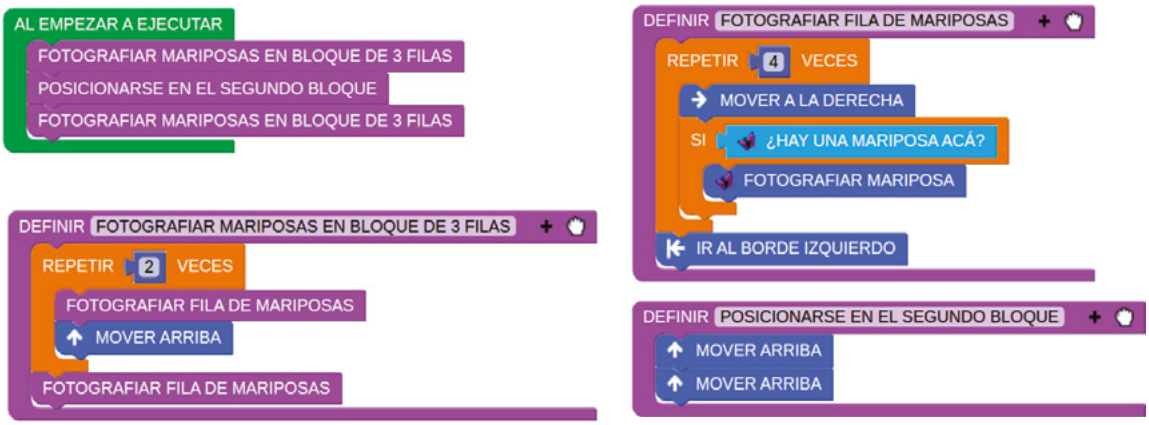
Desarrollo >

El **propósito de este momento** es reforzar el diseño de estrategias legibles que contemplen restricciones impuestas por las primitivas disponibles.

Orientaciones

Invitamos a los grupos a resolver el desafío y modificar la estrategia en caso de ser necesario. Reforzamos la importancia de la legibilidad y el uso de procedimientos para poder reutilizar fragmentos de la solución.

Quienes tengan presente el desafío ***Reparadora de telescopios*** es probable que identifiquen dos bloques de mariposas separados por una fila vacía. También es posible realizar un programa para recorrer todas las filas y que contemple que pueda o no aparecer una mariposa en cualquier casillero del recorrido. En ambos casos, es importante tener en cuenta el tratamiento del caso de borde de la repetición elegida (visto en la secuencia "Programamos estrategias en Pilas Bloques").



Solución posible al desafío ***Mariposas*** en la que el que se trabaja en dos bloques de mariposas separados por una fila vacía.

Otra solución posible al desafío ***Mariposas*** en la que se considera el problema como una sucesión de filas en las que pueden o no aparecer mariposas.

Cierre >

El **propósito de este momento** es recuperar los desafíos de esta secuencia y las anteriores para generar un momento de reflexión y metacognición en el que las y los estudiantes identifiquen en su experiencia de programación los conceptos abordados.

Orientaciones

Para finalizar esta secuencia invitamos a las y los estudiantes a que identifiquen en su experiencia de resolución de los desafíos momentos de aprendizaje y de aplicación de conceptos y herramientas de programación. Nos interesa que entre todo el curso surjan los objetivos de

aprendizaje de esta secuencia y las anteriores. En particular, para esta secuencia, podemos mencionar:

- El análisis de los desafíos para identificar **regularidades** nos permite elaborar estrategias que aprovechen la **reutilización** y la **repetición**. Esto sucedió en todos los desafíos de esta secuencia, pero también en los de otras, por ejemplo, en la secuencia "Programamos estrategias en Pilas Bloques" de esta colección.
- Al incorporar la **alternativa condicional** como herramienta es posible atender problemas con variaciones y, por lo tanto, podemos resolver programando una nueva variedad de problemas. Podemos contrastar con los desafíos de la "Programamos estrategias en Pilas Bloques" y especular qué variaciones de aquellos podríamos haber resuelto con la alternativa condicional.
- Diferenciar aquellas características que se mantienen constantes de aquellas que varían entre las distintas situaciones, casos o instancias es clave para el diseño de una solución para los **problemas variables**. Esto nos permitió elaborar un programa para resolver el problema y, más aún, también nos permitió identificar **subproblemas** que, gracias a que pueden admitir variaciones, podemos encontrarlos más de una vez en el escenario, por ejemplo, las filas con objetos en ***¿Latas o papeles?*** Esta posibilidad es fundamental para reutilizar el fragmento de programa que lo resuelve.
- **Combinar las herramientas de programación** vistas permite resolver una variedad cada vez más amplia de problemas. **Combinar la repetición con la alternativa condicional** nos permitió resolver desafíos con escenarios variables cada vez más grandes (por ejemplo, cuando pasamos de resolver el desafío ***Jugadore de toda la cancha*** a ***Barrilete cósmico*** de la secuencia "Resolvemos recorridos cambiantes" de esta colección). También, el uso de procedimientos nos permitió reutilizar soluciones, facilitando la elaboración y la interpretación de los programas en escenarios con patrones de recorridos, objetos u obstáculos repetidos.

Secuencia didáctica 11

Creamos desafíos cambiantes

Estrategia, repetición y alternativa condicional

¿Cómo son los problemas que requieren el uso de alternativa condicional? ¿Qué información necesitamos para resolverlos? ¿Qué tipo de problemas podemos resolver combinando la alternativa condicional con otras herramientas de programación conocidas?

En esta secuencia, las y los estudiantes crean desafíos en Pilas Bloques que, para ser resueltos, requieren el uso de alternativas condicionales, repeticiones y procedimientos, combinadas de maneras particulares. Como parte de este proceso de creación deberán identificar relaciones entre la aplicación de estas nociones y las características de un desafío.

Actividad 1

Las y los estudiantes analizan los escenarios de un desafío con escenarios cambiantes para identificar qué sensores necesitarían para resolverlo.

Actividad 2

Las y los estudiantes diseñan desafíos que requieran el uso de sensores determinados para reforzar las relaciones entre variaciones en los escenarios posibles y los sensores necesarios para resolverlos.

Actividad 3

Las y los estudiantes diseñan desafíos que requieran una combinación particular de la alternativa condicional con otras herramientas de programación trabajadas.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategias de solución.

Eje: Lenguajes de programación

- Herramientas de lenguaje de programación: repetición simple, procedimientos, alternativa condicional.

Objetivos de aprendizaje

- Asociar las variaciones de un problema (como los escenarios posibles de un desafío) con la información necesaria para resolverlo con un programa (expresada como sensores).
- Identificar, de manera más general o abstracta, relaciones entre la estructura de un problema y la necesidad de utilizar alternativa condicional en un programa que lo resuelva.

Saberes previos de CC

Área: Programación

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategias de solución.

Eje: Lenguajes de programación

- Herramientas de lenguaje de programación: repetición simple, procedimientos, alternativa condicional.

Materiales necesarios

- Dispositivos con Pilas Bloques instalado o acceso a su versión en línea https://pilasbloques.program.ar/

Actividad 1
¿Qué sensores necesitamos?

Al analizar los escenarios de desafíos de alternativa condicional establecemos relaciones entre los escenarios posibles y los sensores disponibles: los **sensores** son la herramienta de programación que nos permite identificar las variaciones de un escenario que podemos aprovechar en una **alternativa condicional** para tomar decisiones sobre la ejecución de ciertos comandos y así resolver en un único programa todas las **variaciones** que propone el desafío[1].

Objetivo >

Se espera que las y los estudiantes:

- Identifiquen a la alternativa condicional y los sensores como herramientas necesarias para resolver un desafío con escenarios cambiantes.
- Establezcan una relación entre los diferentes escenarios posibles de un desafío y los sensores necesarios para resolverlo.

Inicio >

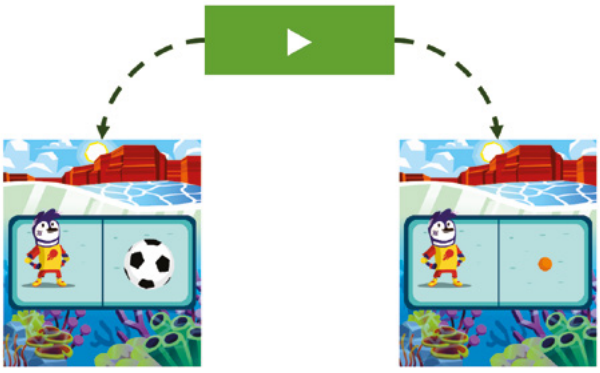
El **propósito de este momento** es recuperar que los desafíos que requieren alternativa condicional para resolverse cuentan con escenarios cambiantes y necesitan que los bloques de sensores estén disponibles.

Orientaciones

Organizamos la clase en grupos heterogéneos pequeños[2]. Para comenzar la primera actividad, observamos y analizamos entre todas y todos el desafío ***¿Pelota o paleta?***, con el que empezamos la secuencia "¿Cómo se resuelven problemas cambiantes?", de esta colección, con el propósito de identificar algunos elementos que caracterizan a los desafíos de esa secuencia.

[1] Esta secuencia requiere que las y los estudiantes estén familiarizados con el Creador de desafíos de Pilas Bloques. Para ello, se recomienda haber trabajado previamente con las secuencias de esta colección "Creamos desafíos de repetición" y "Creamos desafíos de procedimientos", que se encuentran disponibles en el sitio curriculum.program.ar.

[2] Se pueden consultar dinámicas lúdicas para el armado de grupos heterogéneos en "Enseñar computación desde la mirada de la Educación Sexual Integral (ESI)", 10.



Los dos escenarios posibles que aparecen en el desafío *¿Pelota o paleta?*

¿Cuál es el problema a resolver en el desafío? ¿Qué herramientas de programación resultan necesarias para resolverlo? ¿Por qué? ¿Cómo participa cada una de la solución? ¿Qué decisiones de diseño del desafío hacen que esto sea así? ¿Qué cosas podrían cambiar y cuáles no para que siga siendo un desafío que requiere la herramienta de programación que mencionaron para resolverse?

A partir de estas preguntas, queremos que las y los estudiantes recuperen la necesidad de utilizar **alternativa condicional** y **sensores** en el programa para resolver desafíos cambiantes (en este caso, hay varios escenarios posibles). Utilizando los sensores disponibles, es posible obtener información del escenario actual (en este caso, para saber cuál es la pelota que aparece) para establecer, mediante la alternativa condicional, qué instrucciones del programa se deben ejecutar. Puede variar el aspecto del desafío, pero es necesario que haya escenarios posibles con distintas disposiciones (distintos objetos, distinto tamaño, etc.) para que sea necesario utilizar sensores y alternativa condicional para poder resolver todos ellos con un único programa.

En los siguientes momentos de la actividad reforzaremos la relación que existe entre los escenarios posibles y los sensores disponibles de un desafío sobre alternativa condicional.

Desarrollo >

El **propósito de este momento** es reforzar la relación que hay entre los escenarios posibles de un desafío y los sensores que aparecen disponibles.

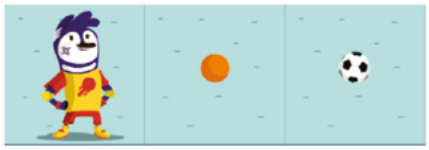
Orientaciones

Al inicio de la actividad, identificamos los elementos más importantes que están presentes en un desafío de alternativa condicional: escenarios cambiantes y sensores. A continuación les anticipamos que, en esta

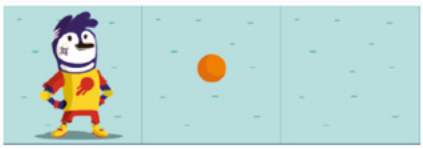
actividad, les vamos a mostrar los escenarios posibles que pueden aparecer en diferentes desafíos y deberán identificar qué sensores son necesarios para poder resolverlos y justificar por qué lo son (en términos de cuáles elementos varían y cuáles no). Para aclarar lo que se pide, vemos entre todos y todas un primer ejemplo.

Si tenemos un desafío con los siguientes escenarios posibles, ¿qué sensores son imprescindibles para resolverlo? ¿Por qué?

Escenario 1



Escenario 2



Les damos unos minutos para que analicen los escenarios y conversen entre sí las respuestas a las preguntas. Luego, hacemos una puesta en común a partir de las respuestas de las y los estudiantes. En este ejemplo, el objetivo es identificar que la pelota de ping-pong (naranja) se mantiene constante en ambos escenarios, mientras que la pelota de fútbol puede estar o no. Por lo tanto, para la pelota de ping-pong, no es necesario utilizar un sensor, pero para la pelota de fútbol, sí, debido a que necesitamos verificar la presencia o ausencia de este elemento cuando ejecutamos el programa (por ejemplo, a partir de un sensor *¿Hay una pelota de fútbol acá?*) para saber si hay que realizar la acción de patear o no.

A continuación, les proponemos que se organicen en grupos[3]. Presentamos otro desafío con sus escenarios posibles y un conjunto de sensores para que analicen en cada grupo. Les recordamos que la consigna es identificar qué sensores son imprescindibles para resolver el desafío y por qué. Es decir, seleccionar un subconjunto mínimo de sensores que permitan resolver el desafío.

[3] Se pueden consultar dinámicas lúdicas para el armado de grupos heterogéneos en "Enseñar computación desde la mirada de la Educación Sexual Integral (ESI)", 10.

Conjunto A de escenarios posibles

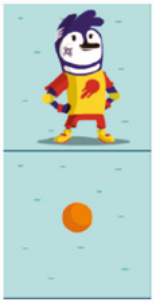
Escenario 1

Escenario 2

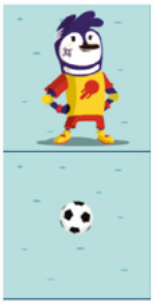
Sensores necesarios[4]

Conjunto B de escenarios posibles

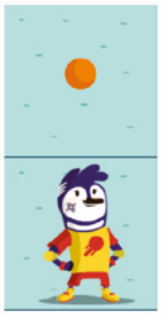
Escenario 1



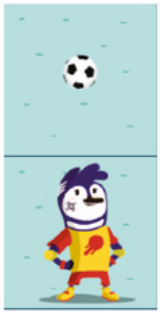
Escenario 2



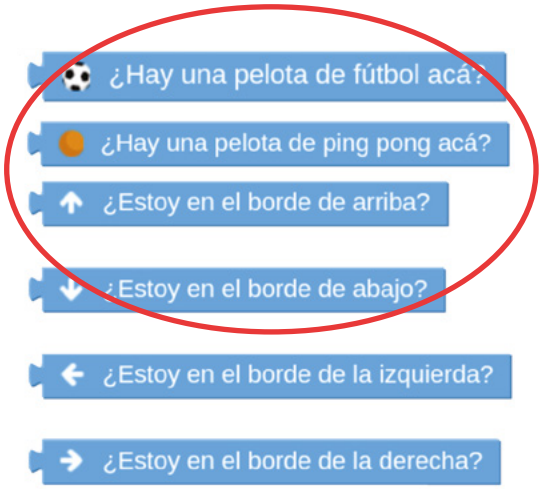
Escenario 3



Escenario 4



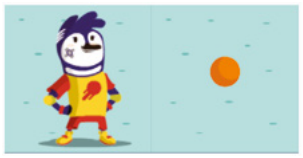
Sensores necesarios



[4] Se dan soluciones posibles para orientación de las y los docentes.

Conjunto C de escenarios posibles	Sensores necesarios

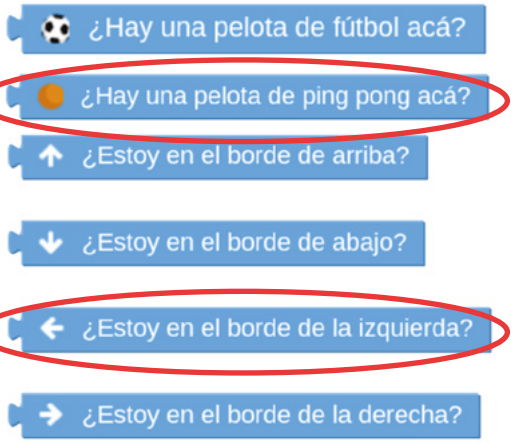
Escenario 1



Escenario 2

Escenario 3

Escenario 4



Mientras los grupos resuelven la actividad, vamos observando las soluciones que proponen y las justificaciones que desarrollan. En caso de que elijan sensores que no son necesarios, podemos aprovechar para reforzar la relación entre los escenarios posibles, los elementos variables y los sensores. Algunas preguntas que pueden servir de guía son las siguientes.

¿Qué elementos están involucrados en el desafío? ¿Cuáles se mantienen constantes y cuáles varían? ¿Tenemos que tomar algún tipo de decisión para los que se mantienen constantes? ¿Por qué?

Luego de que los grupos hayan acabado, proponemos una ronda en la que cada grupo presente alguno de los desafíos y comparta los sensores que eligió y **las justificaciones que registraron**. En el caso de que otros grupos quieran realizar aportes, habilitamos un espacio para el intercambio. Como conclusiones, podemos mencionar casos de sensores que fueron considerados y finalmente no hayan sido necesarios, otros

escenarios que podrían agregarse al desafío para ser resueltos con los mismos sensores o bien, reconocer sensores que podrían agregarse para resolver otras variantes de los escenarios.

Cierre >

El **propósito de este momento** es reforzar la relación entre los sensores y los escenarios cambiantes a partir de analizar desafíos existentes de alternativas condicionales.

Orientaciones

Para reforzar la necesidad de contar varios escenarios posibles y sensores que permitan identificar esas variaciones, les proponemos analizar en grupos otros dos desafíos de alternativa condicional de Pilas Bloques que trabajamos en tres secuencias de esta colección: "¿Cómo se resuelven problemas cambiantes?", "Resolvemos recorridos cambiantes" y "Programamos estrategias para problemas cambiantes".

Para guiar el análisis de los desafíos, les sugerimos que cada grupo responda las siguientes preguntas:

- ¿Qué elementos varían en cada uno de los escenarios? ¿Cuáles se mantienen constantes?
- ¿Qué relación hay entre los escenarios posibles y los sensores disponibles? ¿Por qué son necesarios esos sensores? ¿Qué sucedería si no tuvieran disponible alguno de esos sensores?
- Piensen otros escenarios del desafío que se podrían resolver con los mismos sensores y comandos disponibles.

A modo de cierre, hacemos una puesta en común de las respuestas que elaboró cada grupo para explicitar la relación entre los sensores disponibles en un desafío y las variaciones que presentan los escenarios del desafío.

- Desde el punto de vista de los **sensores**: en un desafío con múltiples escenarios, tienen que estar disponibles aquellos que nos permitan **obtener la información que necesitamos para resolver todos los casos posibles que plantean los escenarios**. Por ejemplo, en el desafío ***Barrilete cósmico***, los sensores nos permiten obtener información sobre dónde está la próxima casilla a la que debemos avanzar; en el desafío ***Festín astronómico***, los sensores nos permiten obtener información sobre el contenido de cada celda para realizar la acción correspondiente (como *observar planeta* u *observar estrella*).
- Desde el punto de vista del diseño de los **escenarios posibles**: debemos considerar **variaciones que puedan ser identificadas con los sensores disponibles** y se puedan combinar con otros comandos para resolver el desafío propuesto. Por ejemplo, en el desafío ***Barrilete cósmico***, podríamos agregar nuevas formas del recorrido siempre y cuando se avance hacia abajo o hacia la derecha (y no, por ejemplo, a la izquierda, dado que no contamos con un sensor para detectar casillas a la izquierda).

Actividad 2
Nuestro desafío de Pilas Bloques para trabajar alternativa condicional

Objetivo >

Se espera que las y los estudiantes:

- Identifiquen las decisiones de diseño necesarias para crear un desafío que involucre sensores y alternativas condicionales.
- Diseñen e implementen desafíos en Pilas Bloques que requieran el uso de la alternativa condicional para resolverse.

Inicio >

El **propósito de este momento** es explicitar que, para que un desafío de Pilas Bloques requiera el uso de la alternativa condicional para resolverse, es necesario que cuente con escenarios cambiantes y tenga como bloques disponibles sensores y comandos de alternativa condicional.

Orientaciones

Para comenzar la actividad, compartimos el objetivo de la actividad con las y los estudiantes: *deberán crear desafíos de Pilas Bloques que requieran el uso de alternativa condicional para resolverse*.

Luego, les proponemos recuperar alguna experiencia previa en la creación de desafíos en Pilas Bloques[5] e identificar decisiones que tuvieron que tener en cuenta para la aplicación de una herramienta o noción particular

[5] En las secuencias previas que abordan el uso del Creador de Desafíos, las y los estudiantes se involucraron en el uso de las herramientas de programación para pensar en su aplicación para solucionar diferentes situaciones problemáticas. De esta manera, el foco del aprendizaje está puesto en la pregunta: *¿cómo debe ser un problema para que su solución involucre el uso de una herramienta de programación específica?* En esta secuencia, aprovecharemos el Creador de desafíos para crear desafíos con escenarios variables cuyas soluciones requieran el uso de alternativas condicionales.

de programación. Por ejemplo, si ya realizamos la secuencia "Creamos desafíos de repetición" o "Creamos desafíos de procedimientos" de esta colección, podemos hacerlo a partir de la pregunta: *¿Qué decisiones de diseño tuvieron que tener en cuenta para crear desafíos que se resuelvan utilizando repetición (o procedimientos)?* En las respuestas buscamos revelar evidencias sobre las decisiones de diseño que tuvieron en cuenta, por ejemplo, los patrones en la ubicación de los objetos y los obstáculos o en la forma del escenario, similitudes/diferencias entre diferentes desafíos, etc. Además, aprovechamos esta instancia para repasar qué elementos básicos del Creador de Desafíos usaron para diseñar los desafíos, por ejemplo, los objetos y la disposición, las dimensiones del escenario, la consigna y los comandos disponibles.

Luego de este momento, retomamos los desafíos analizados en la **Actividad 1** con el fin de identificar cuáles son las nuevas decisiones que deberán tener en cuenta para diseñar un desafío de alternativa condicional.

*Piensen en los desafíos que analizaron en la **Actividad 1** y piensen que van a tener que crear un desafío que requiera alternativa condicional para resolverse. ¿Qué va a tener que tener ese desafío?*

Buscamos que arriben a las siguientes conclusiones:

- Es necesario agregar varios **escenarios iniciales posibles**. Como empezamos a observar en la **Actividad 1**, podemos pensar este conjunto de escenarios como un escenario con variaciones particulares (por ejemplo, si en una casilla aparece una pelota de ping-pong o de fútbol).
- Hay que seleccionar y poner a disposición **comandos de alternativa condicional y sensores** que permitan obtener información necesaria del escenario para resolver con un único programa todos los escenarios posibles. Por ejemplo, si va a variar el tipo de pelota, necesitamos sensores que capturen esa información.

Desarrollo >

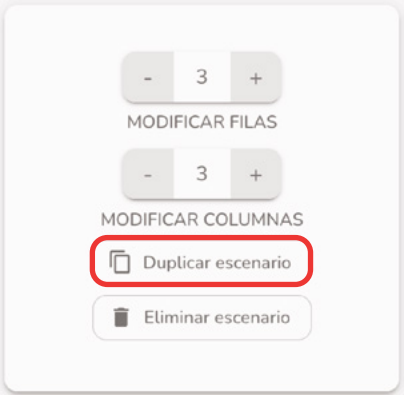
El **propósito de este momento** es que las y los estudiantes identifiquen qué características debe tener un desafío de Pilas Bloques para que su solución requiera el uso de alternativas condicionales.

Orientaciones

Organizamos la clase en grupos heterogéneos pequeños y les indicamos que ingresen al Creador de desafíos de Pilas Bloques y elijan un personaje. Antes de repartir las consignas y comenzar a trabajar

recordamos cuestiones instrumentales básicas (cómo elegir el personaje, definir el tamaño del escenario y agregar objetos y obstáculos). Retomamos, de las conclusiones del inicio, la necesidad de definir escenarios alternativos e incorporar sensores a los bloques disponibles.

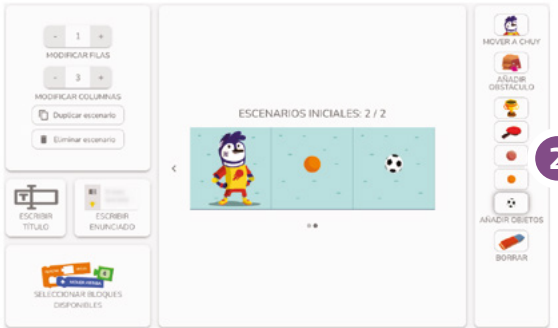
Para **incorporar escenarios alternativos** mostramos la función del botón "Duplicar escenario", debajo de los controles de las dimensiones del escenario.



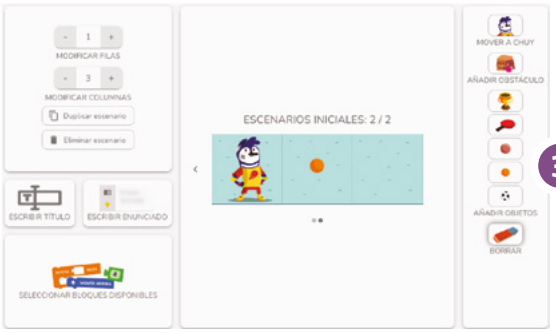
El botón "Duplicar escenario" junto a los controles del tamaño del escenario.

Este botón agrega como un escenario posible del desafío una copia del escenario actual para que le incorporemos variaciones. Por ejemplo, para crear los dos escenarios que analizamos al comienzo del desarrollo de la **Actividad 1**, podemos crear el primero con las dos pelotas, duplicar el escenario y borrar la pelota de fútbol del segundo escenario.

1 Creamos el primer escenario, estableciendo la dimensión y agregando ambas pelotas.



2 Creamos el segundo escenario duplicando el primero. Vemos que estamos en el segundo escenario de dos posibles (por eso el título aclara 2/2 y vemos los dos puntos debajo del dibujo del escenario); podemos cambiar de uno a otro presionando las flechas que aparecen a la izquierda o a la derecha del dibujo del escenario.



3

Utilizamos la herramienta "Borrar" para eliminar la pelota de fútbol y obtener el escenario que necesitábamos.

Para incorporar sensores lo hacemos desde la selección de los bloques disponibles. Buscamos la categoría "Sensores" y activamos solo los que vayamos a usar. En este caso, sería *¿Hay una pelota de fútbol acá?*

Botón para seleccionar los bloques disponibles al resolver el desafío.

Se verán todos los sensores disponibles, para elegir cuáles se quiere que aparezcan disponibles al resolver el desafío.

Resueltas las cuestiones instrumentales, entregamos a cada grupo una tarjeta con una consigna sin que la vean los demás grupos (Ver Anexo), porque la dinámica de la actividad consistirá en que otro grupo resuelva el desafío por crear.

Tarjeta 1	Tarjeta 2	Tarjeta 3
Crear un desafío que se resuelva con un programa que utilice alternativa condicional y un *sensor de objeto*, por ejemplo: ¿Hay una paleta acá*?*	Crear un desafío que se resuelva con un programa que utilice alternativa condicional y un *sensor de obstáculo*, por ejemplo: ¿Hay un obstáculo a la derecha?	Crear un desafío que se resuelva con un programa que utilice alternativa condicional y un *sensor de borde*, por ejemplo: ¿Estoy en el borde de arriba?

Consignas para los desafíos de esta actividad.

Alentamos a documentar las decisiones que van tomando, por ejemplo, con las siguientes preguntas para que respondan. Estas preguntas serán insumo para realizar una puesta en común en el cierre de la actividad.

- ¿Qué sensor/es eligieron? ¿Qué hicieron que varíe entre los distintos escenarios? ¿Cómo relacionan esta decisión con el sensor que tuvieron que utilizar?
- ¿Qué otros bloques pusieron como disponibles? ¿Por qué?
- ¿En qué orden fueron tomando las decisiones? ¿Qué fue lo primero que tuvieron en cuenta y qué lo último?

Recorremos los grupos mientras trabajan en la creación del desafío, prestando atención a las inquietudes y los avances. Es probable que necesitemos acompañar el trabajo de cada grupo individualmente, especialmente para ayudarlos a identificar cuáles son las características clave que deben incorporar al desafío para cumplir con el objetivo pedido.

Algunas recomendaciones para ordenar el trabajo son las siguientes:

- Sugerir que **no comiencen diseñando escenarios complejos** (para que la creación y la solución del desafío requieran el menor tiempo posible) y que no haya demasiados posibles. Podemos sugerir también que realicen una primera versión solo con un sensor y dos escenarios.
- Orientarlos con preguntas sobre cómo deben ser los escenarios de los desafíos para cumplir con los requisitos de la tarjeta. Podemos recomendarles que comiencen por elegir el o los sensores para cumplir con la tarjeta y luego qué objetos y obstáculos van a estar involucrados y cómo van a variar. Luego, podemos plantearles: ¿Qué escenarios posibles se imaginan? ¿Qué comandos o primitivas van a

necesitar? ¿Van a necesitar otros comandos (como procedimientos o repetición)?

- Proponer a los grupos que no sepan cómo comenzar que escriban primero un **boceto** de la forma del programa que cumpla con el objetivo de la tarjeta para avanzar hacia una solución abierta, pero más concreta sobre la que seguir trabajando.
- Recordar con los grupos que pueden consultar desafíos de otras secuencias cuyas soluciones cumplan con los requisitos de la tarjeta que les tocó. Les mencionamos que no buscamos replicar un desafío, sino utilizarlo como inspiración.

Al finalizar, intercambiarán los desafíos para que sean resueltos por un grupo diferente al que los creó y que haya trabajado con una tarjeta diferente. Esto se puede **realizar cuando consideremos que el desafío está listo, pero también se puede alentar como una iteración que sea parte del proceso de creación** si se detectan dificultades en el diseño mientras se acompaña el trabajo de los grupos. Por eso, es importante que los grupos arriben rápido a una primera versión del desafío que se pueda probar (tanto por ellas y ellos mismos como por otro grupo). Es importante evaluar en estas instancias si el desafío propuesto tiene solución y, en caso de que así sea, si la solución que elaboró el otro grupo cumple con la consigna de la tarjeta. Para agilizar el trabajo podemos proponer trabajar en intervalos de tiempo determinados (por ejemplo, diez minutos).

En cada iteración, promovemos la devolución al grupo creador para motivar reflexiones que permitan asociar la disposición del escenario y los bloques disponibles del desafío con las soluciones posibles. Podemos hacer preguntas que los inviten a hipotetizar variaciones del desafío, identificar dificultades o errores y explicitar los razonamientos que hicieron en el proceso de creación. Recomendamos que cada grupo tome nota de las devoluciones que le han dado para que puedan hacer los cambios pertinentes en una instancia posterior.

Cierre >

El **propósito de este momento** es abordar una instancia de metacognición por parte de las y los estudiantes para identificar que en el proceso de construcción de los desafíos fue necesario descubrir relaciones entre los escenarios posibles, los bloques disponibles de sensores y de alternativa condicional y las características del programa que lo resuelve.

Orientaciones

El objetivo de esta instancia es habilitar un espacio de puesta en común para presentar y analizar los desafíos que diseñaron. Invitamos a los grupos a que compartan sus desafíos y que, a partir de las notas de su proceso y la devolución recibida, compartan las decisiones que tomaron a lo largo de la creación del desafío.

Nos interesa reparar en que la complejidad de la **Actividad 2** reside en encontrar una relación entre los escenarios posibles, los sensores disponibles y las posibles soluciones. Es importante rescatar cómo fue construyendo esta relación cada uno de los grupos. Por ejemplo, podemos identificar que si habían decidido que la solución utilice el bloque "¿Hay una pelota de fútbol acá?", tuvieron que poner ese sensor como disponible y diseñar escenarios posibles que varíen según dónde aparecen las pelotas de fútbol.

Podemos mencionar que la práctica que hicieron es similar a la que hacen cuando resuelven un desafío ya creado (es decir, buscar relaciones entre el problema que tienen que resolver y las herramientas de programación que van a utilizar para resolverlo) pero que hacerlo "al revés" permitió concentrarse en la relación entre esos elementos y, por lo tanto, forma parte de una instancia más profunda de su aprendizaje de programación.

Para explicitar las relaciones encontradas a partir de cada tarjeta e intentar expresarlas de manera más general, podemos partir de la comparación de desafíos que cumplan con el objetivo con otros que no. Podemos considerar tanto las versiones finales de los desafíos como las intermedias (lo que nos permite recuperar los motivos por los cuales los desafíos tuvieron que ser mejorados) o desafíos de ejemplo que no formen parte de la producción de los grupos.

Dos conjuntos de escenarios posibles: a la izquierda, uno que cumple con la consigna de requerir un sensor de obstáculo (Tarjeta 2) pero no de objeto (Tarjeta 1), ya que los obstáculos varían entre un escenario y otro pero no los objetos; a la derecha, la situación opuesta.

Actividad 3
Nuestro desafío integrador

Objetivo >

Se espera que las y los estudiantes:

- Diseñen desafíos de programación que se resuelvan combinando alternativas condicionales con procedimientos y repeticiones.
- Identifiquen las características que tienen que tener los escenarios de un desafío a partir de un programa dado con una estructura determinada.

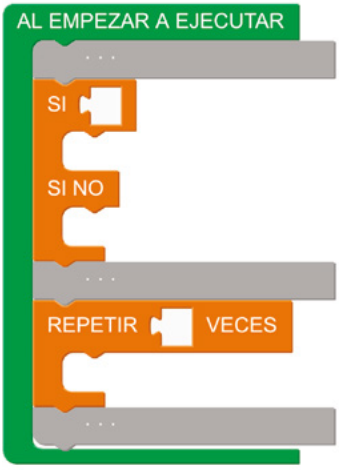
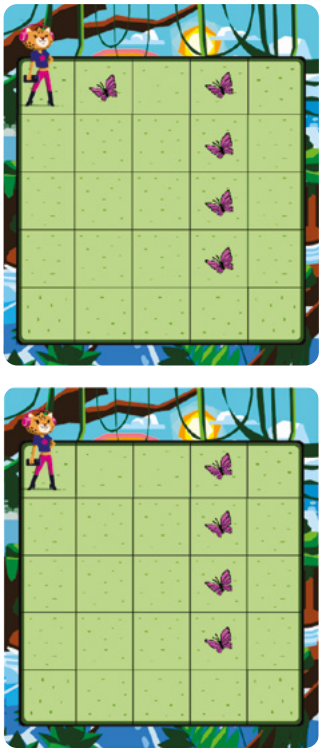
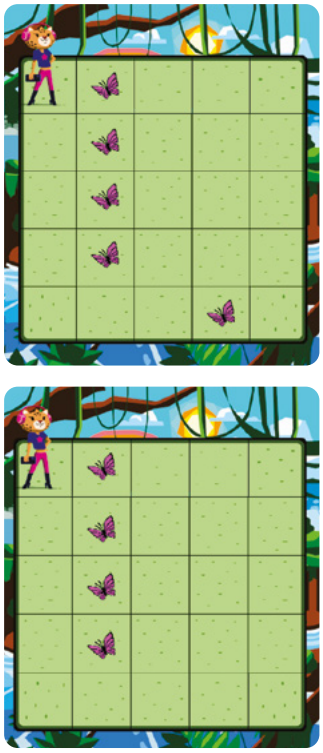
Inicio >

El **propósito de este momento** es presentar una manera de expresar la estructura que debe tener un programa (en términos de cómo se combinan los bloques de alternativa condicional y de repetición) y exhibir que existe una relación entre esta estructura y la disposición de los escenarios posibles de un desafío que se resuelva con ese programa.

Orientaciones

Comenzamos con una actividad desenchufada para introducir el problema de asociar configuraciones de los escenarios con la estructura de los programas de las soluciones (en particular, cómo se combina la alternativa condicional con la repetición).

En este caso, mostraremos una estructura de programa y dos conjuntos de escenarios para que las y los estudiantes decidan cuál de estos conjuntos podría ser resuelto con un programa que cumpla con la estructura pedida. Para argumentar la decisión pueden proponer una estrategia de solución para resolver el desafío hipotético y mostrar que esa estrategia respeta la estructura pedida. Vemos la estructura y los escenarios a continuación.

Estructura del programa	Escenarios posibles A	Escenarios posibles B
 *Primero, una alternativa y después una repetición.*		

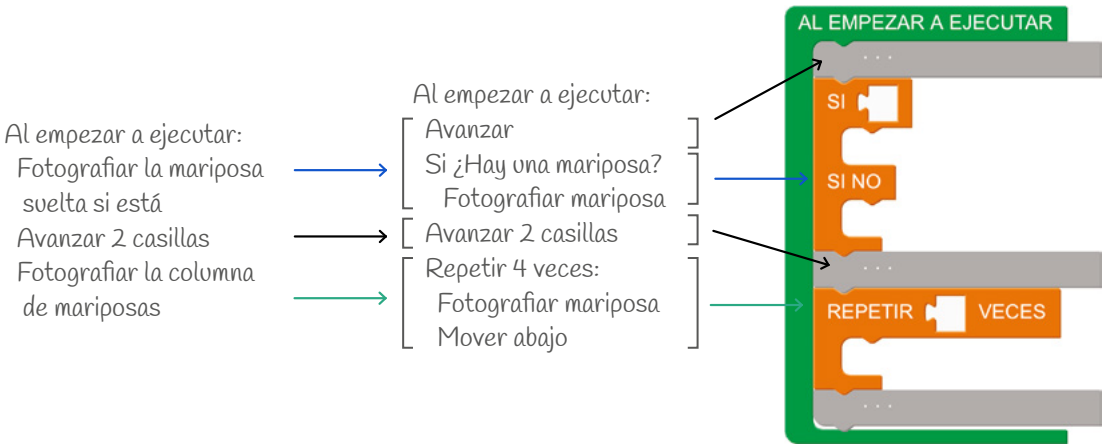
Al presentar la actividad es importante reforzar que:

- Los **bloques grises** representan partes del programa donde puede haber otros bloques o, incluso, no haber bloques.
- **Buscamos que la solución para el desafío que diseñarán respete la estructura dada en la tarjeta** (es decir, no se pueden quitar o reordenar los bloques de repetición o alternativa).
- Alguna rama de la alternativa (el caso SI o el caso SI NO) **puede quedar vacía** (esto nos permite no considerar las diferencias entre los casos en los que se requiere el uso de SI del uso del SI-SI NO).

Para argumentar por qué elegir los bloques A[6], primero tendrán que reconocer que es necesario atender a la situación cambiante (la mariposa que a veces está y a veces no). Para ello, se requiere el uso de la

[6] Existe una solución al conjunto de escenarios B que respeta la estructura pedida, pero puede que sea menos evidente para las y los estudiantes. Consiste en recorrer el escenario hasta la casilla donde puede aparecer la mariposa, evaluar si la mariposa está, fotografiarla si está y luego regresar para fotografiar la hilera de mariposas. Más allá de la solución que programen, lo más importante de este momento es la justificación que construyan de la relación entre estrategias y estructuras de programas.

alternativa condicional y luego resolver la acción repetitiva (fotografiar la columna de mariposas), para la que se requiere el uso de la repetición simple. Esto resulta evidente al escribir la estrategia para un programa sin procedimientos. Si hubieran utilizado procedimientos para expresar la estrategia, podemos realizar la justificación sobre una versión sin procedimientos equivalente. Es importante que se observe que en el programa o estrategia propuesto siempre se ejecuta la alternativa condicional antes de la repetición simple, independientemente de cómo estén definidos los procedimientos.



Una estrategia posible para resolver un desafío con los escenarios del conjunto A y su relación con la estructura pedida. Primero, una versión con procedimientos y luego su equivalente sin procedimientos.

Invitamos a los grupos a compartir sus justificaciones con los demás y entender cómo cada uno relacionó los escenarios elegidos con la estructura pedida y por qué las estrategias propuestas la respetan.

Desarrollo >

El **propósito de este momento** es que las y los estudiantes reconozcan características de los conjuntos de escenarios que requieren que los programas que los resuelven tengan una estructura determinada (que combine la alternativa condicional con las otras de herramientas de programación trabajadas hasta el momento).

Orientaciones

Estructuramos la actividad como una serie de partes, todas con la misma dinámica: cada grupo recibe una estructura esperada y crea un desafío que requiere de un programa con esa estructura para resolverlo. Luego, se pone a prueba: lo intercambian con otro grupo para que lo

resuelvan. Presentamos las partes en un orden creciente de dificultad para ir planteándolas en función de los avances y los obstáculos que observamos.

Parte 1: se trabaja con programas que combinan alternativas condicionales con repetición con distintas formas de anidamiento.

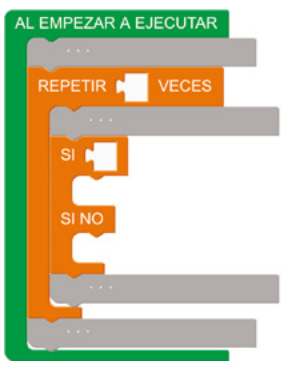
Respecto al modo de trabajo, los tiempos y la puesta a prueba en repetidas iteraciones, se puede replicar la misma dinámica que se haya definido en la **Actividad 2**. Al igual que en esa actividad, luego de elaborar el desafío, se pasará el desafío a otro grupo para que lo resuelva en un tiempo acordado (por ejemplo, 10 minutos). Al finalizar, haremos un intercambio entre los grupos para analizar las soluciones y evaluar si cumple o no con la estructura dada en la tarjeta.

Le entregamos a cada grupo una tarjeta con la estructura esperada (Ver Anexo). Es importante enfatizar que el desafío debe poder ser resuelto en el tiempo acotado definido. Para ello, deberán prestar atención a la cantidad de escenarios posibles, que los bloques que se ponen a disposición sean solo los que se necesitan y que la consigna sea clara.

Recordamos que los bloques grises indican que allí pueden colocarse bloques cualesquiera o, incluso, ninguno y que no diferenciaremos entre los bloques Si y Si-Si no (es decir, el caso para Si no puede quedar vacío).

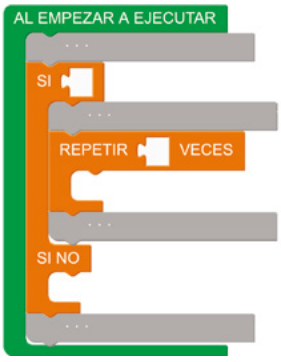
Podemos introducir la relación entre el uso de la repetición y la disposición de los objetos en el escenario recuperando lo trabajado en la secuencia didáctica de esta colección "Creamos desafíos de repetición".

Tarjeta 1



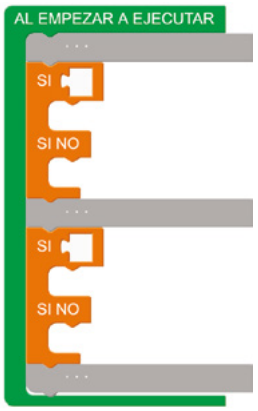
Una alternativa adentro de una repetición.

Tarjeta 2



Una repetición adentro de una alternativa.

Tarjeta 3



Una alternativa después de otra.

Tarjetas con las estructuras pedidas para los programas de esta parte.

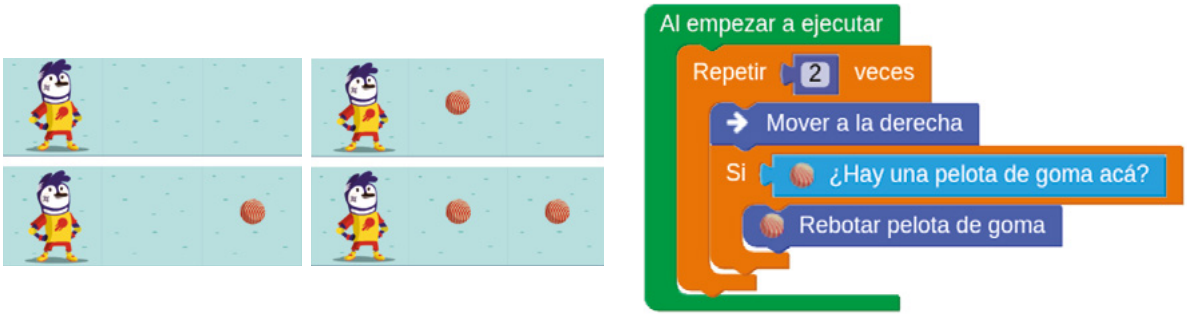
Además de diseñar el desafío, deberán responder a las siguientes preguntas para documentar el proceso:

- Además de los bloques del programa de referencia, ¿qué otros bloques deberían estar disponibles para resolver el desafío? ¿Por qué?
- ¿Cómo relacionarían las variaciones en el escenario con las alternativas condicionales que están presentes en el programa?

Al finalizar cada parte, además de compartir los desafíos, invitamos a los grupos a compartir el programa de la tarjeta entregada y las decisiones que fueron tomando en la elaboración para que puedan socializar la justificación de sus elecciones. Nuevamente, buscamos **reforzar la relación entre la estructura de una solución y el diseño del desafío**. Esto puede involucrar aspectos como el diseño de los escenarios posibles, los sensores y comandos disponibles y la formulación de la consigna.

¿De qué manera se relaciona la estructura pedida en la tarjeta 1 con la disposición de los elementos en los escenarios posibles del desafío que diseñaron?

Para identificar estas relaciones podemos comparar desafíos que responden a la misma consigna y contrastarlos con los que responden a las otras. Presentamos un ejemplo a continuación.



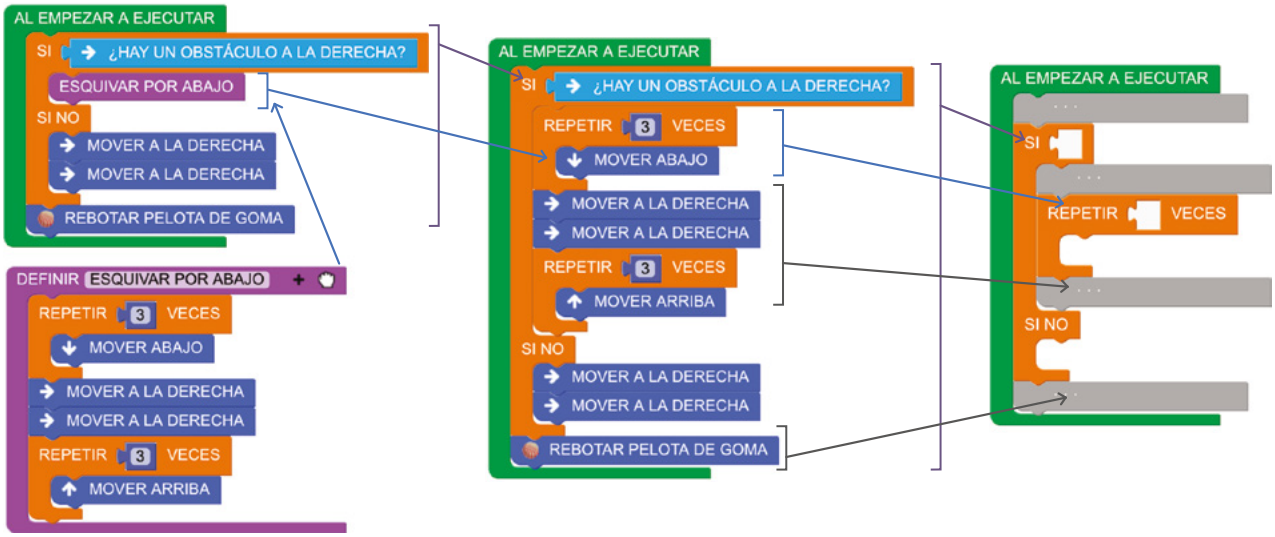
Una solución posible para la Tarjeta 1: los escenarios posibles y un programa que lo resuelve y cumple con la estructura pedida en la tarjeta.

Como estamos ejecutando la alternativa dentro de una repetición, esto es una evidencia de que debemos tomar una decisión reiteradas veces a partir de un mismo sensor. En este caso, elegimos armar un conjunto de escenarios posibles en los que hay una sucesión de casillas en las que puede o no haber una pelota para rebotar.

Para contrastar esta situación, podemos invitar a un grupo que haya resuelto la **tarjeta 2** a compartir su desafío. En el siguiente ejemplo,

presentamos los escenarios posibles y la justificación a partir de un programa que lo resuelve (que usa procedimientos).

Escenarios posibles de un desafío que cumple con la tarjeta 2.

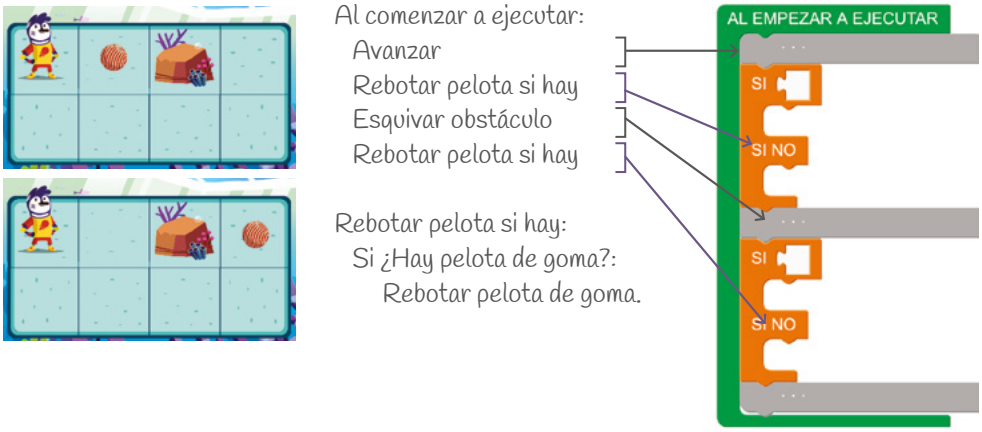


Un programa con procedimientos que resuelve el desafío planteado (izquierda) y su relación con la estructura pedida (derecha) . La estructura se vuelve más visible en una versión equivalente sin procedimientos (centro).

¿De qué manera se relaciona la estructura pedida en la tarjeta 2 con la disposición de los elementos en los escenarios posibles de los desafíos que crearon para resolverla? ¿En qué se diferencia de las otras estructuras de esta parte?

En este caso, dado que la alternativa no está dentro de una repetición, el desafío debe poder resolverse tomando una única decisión (en este caso, si está ocupado o libre el casillero de la derecha) y después avanzar sin volver a consultar información sobre el escenario. Podemos contrastar esta situación con la planteada por la tarjeta 1 en la que el desafío requería tomar una decisión repetidas veces.

Invitamos a los grupos que trabajaron con la **tarjeta 3** a que compartan sus soluciones y justificaciones para contrastar esta situación. Vemos un ejemplo a continuación.



Los escenarios posibles de un desafío que responde a la tarjeta 3 y una justificación a partir de una estrategia de solución con procedimientos.

¿De qué manera se relaciona la estructura pedida en la tarjeta 3 con la disposición de los elementos en los escenarios posibles de los desafíos que crearon para resolverla? ¿En qué se diferencia de las otras estructuras de esta parte?

La estructura de la **tarjeta 3** requiere de un desafío que contiene dos momentos en los que tiene que tomarse una decisión puntual. Si, además, queremos que no se pueda resolver con una repetición (que sería el caso de la **tarjeta 1**), estas dos situaciones deben estar separadas una de otra, por un subproblema que no pueda resolverse con la misma alternativa.

Parte 2: nos centraremos en las estrategias que incluyen alternativas condicionales. Para eso, repartiremos tarjetas que requieren desafíos que combinen alternativas con procedimientos.

Para recuperar algunas ideas sobre el diseño de desafíos que requieran procedimientos podemos apelar al trabajo con la secuencia didáctica de esta colección "Creamos desafíos de procedimientos". Para recuperar la combinación de procedimientos con alternativa condicional podemos recuperar el trabajo con las secuencias "¿Cómo se resuelven problemas cambiantes?" y "Programamos estrategias para problemas cambiantes".

Tarjeta 1

Un procedimiento al principio, una alternativa con un procedimiento y un procedimiento al final.

**Pueden agregar algunas primitivas por fuera de los procedimientos si hace falta.*

**Pueden usar el mismo procedimiento más de una vez.*

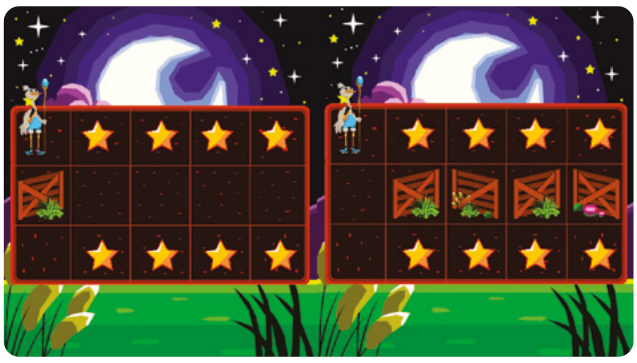
Tarjeta 2

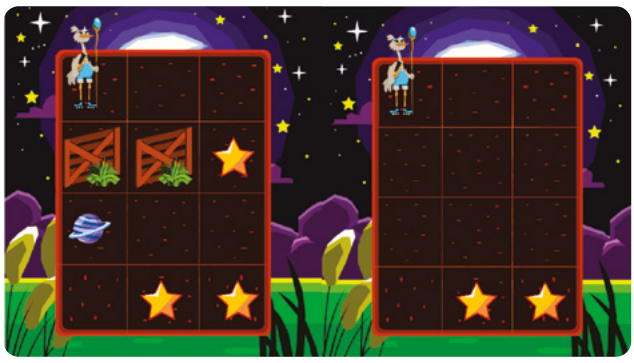
Primero, una alternativa con dos procedimientos. A continuación, otro procedimiento.

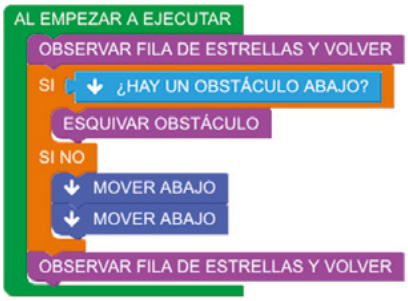
**Pueden agregar algunas primitivas por fuera de los procedimientos si hace falta.*

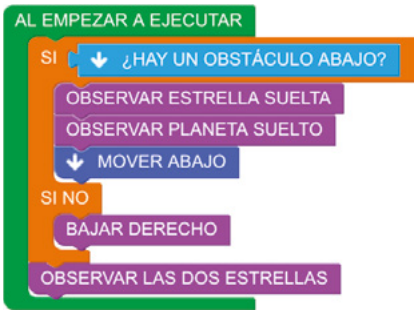
**Pueden usar el mismo procedimiento más de una vez.*

Al igual que en la Parte 1, realizamos una puesta en común en la que invitamos a los grupos a compartir sus desafíos y las decisiones que tomaron, explicando con sus palabras cómo el diseño del escenario se relaciona con la estructura pedida en la tarjeta y con la noción de estrategia y de división en subproblemas.







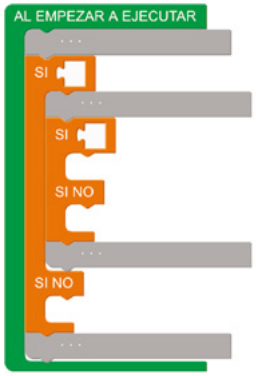


Escenarios posibles y programas (fragmentos) que los resuelven para las tarjetas 1 y 2 de esta parte.

Después de que hayan completado el trabajo, invitamos a los grupos a compartir sus desafíos para compararlos y contrastarlos, para identificar relaciones generales entre las disposiciones del escenario y la estructura pedida de los programas. En el ejemplo, vemos que el primer desafío plantea un problema con tres subproblemas diferenciados, uno después de otro, de los cuales el segundo es cambiante (cómo pasar a la última fila) y por eso involucra la alternativa condicional. En el caso de la segunda tarjeta, la parte cambiante del desafío involucra dos subproblemas y por eso, aparecen dos procedimientos dentro de la alternativa condicional.

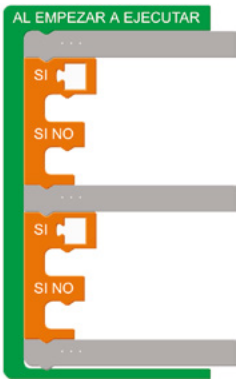
Parte 3: proponemos consignas que apuntan a establecer relaciones entre las formas de anidamiento de los bloques y las estructuras de los escenarios.

Tarjeta 1



Una alternativa condicional anidada dentro de otra.

Tarjeta 2



Una alternativa condicional a continuación de la otra.

Escenarios posibles de desafíos que resuelven la tarjeta 1 (izquierda) y 2 (derecha), y programas para resolverlos.

Al igual que en las partes 1 y 2, invitamos a compartir los desafíos para identificar semejanzas y diferencias con la intención de generalizar las características de los escenarios propios de cada tarjeta.

Las alternativas anidadas implican una situación cambiante dentro de otra. En el ejemplo, tenemos un recorrido que debe cambiar según si aparece la tranquera o no. Además, como parte de ese recorrido, puede aparecer un planeta para observar o no. En cambio, las alternativas secuenciales implican dos situaciones cambiantes pero independientes. En este caso, por un lado, avanzar hasta el objetivo evitando el obstáculo según dónde aparezca y luego, por otro, observar el astro que corresponda.

Cierre >

El **propósito de este momento** es brindar una instancia de metacognición para identificar que en el proceso de construcción de los desafíos fue necesario descubrir relaciones entre la disposición de los escenarios con los bloques disponibles y la estructura del programa que lo resuelve y, también, para explicitar estas relaciones y generalizarlas, teniendo en cuenta la importancia del análisis del problema a la hora de proponer un programa para solucionarlo.

Orientaciones

Para cerrar esta secuencia, invitamos a las y los estudiantes a identificar situaciones generales en los desafíos que crearon en las sucesivas partes. Nos interesa recuperar y reforzar las reflexiones que realizamos en las puestas en común, asociando estructuras de programas y características de los escenarios posibles. Si, al analizar un problema, identificamos alguna de estas características, ya sabemos qué estructura tendrá una posible solución, y puede ser un paso para comenzar a resolverlo. Por ejemplo, podemos enunciar: "Cuando aparece una misma situación variable, una a continuación de la otra, podemos usar una repetición con una alternativa anidada para resolverlo". Este tipo de reflexiones ya pueden haber surgido si resolvimos las secuencias "¿Cómo se resuelven problemas cambiantes?", "Resolvemos recorridos cambiantes" y "Programamos estrategias para problemas cambiantes", de esta colección. Es valioso recuperarlas y pensar en cómo se pusieron en juego para la tarea de crear desafíos y cómo esta tarea ayudó a reforzarlas.

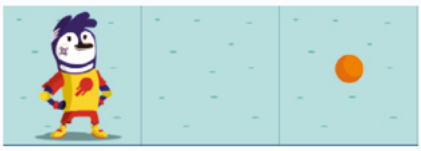
Para terminar, podemos preguntar explícitamente: *¿En qué medida estas actividades les ayudaron a afianzar o aclarar ideas sobre el uso de alternativas condicionales? ¿Pueden identificar momentos de la actividad en los que hayan aprendido algo nuevo o reforzado algo que no tenían tan claro?*

Alentamos que las y los estudiantes analicen su experiencia en clave de metacognición para reforzar uno de los objetivos principales de esta secuencia: reforzar sus habilidades de programación, en vez de resolviendo más desafíos, reflexionando sobre cómo debe ser un desafío para que su solución tenga determinadas características.

Anexo

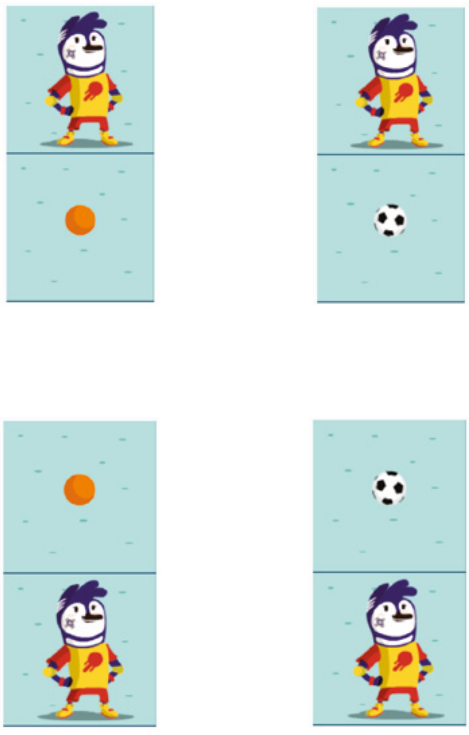
Actividad 1

Conjunto A de escenarios posibles



Sensores necesarios

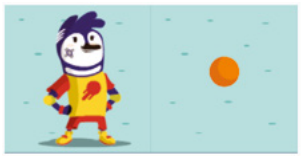
Conjunto B de escenarios posibles



Sensores necesarios

Conjunto C de escenarios posibles	Sensores necesarios

Escenario 1



Escenario 2

Escenario 3

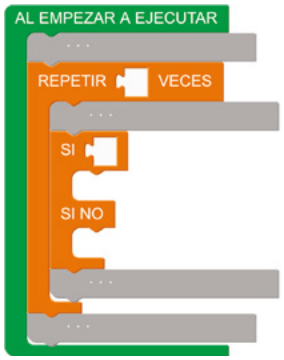
Escenario 4

Actividad 2

Tarjeta 1	Tarjeta 2	Tarjeta 3
Crear un desafío que se resuelva con un programa que utilice alternativa condicional y un *sensor de objeto*, por ejemplo: ¿Hay una paleta acá?	Crear un desafío que se resuelva con un programa que utilice alternativa condicional y un *sensor de obstáculo*, por ejemplo: ¿Hay un obstáculo a la derecha?	Crear un desafío que se resuelva con un programa que utilice alternativa condicional y un *sensor de borde*, por ejemplo: ¿Estoy en el borde de arriba?

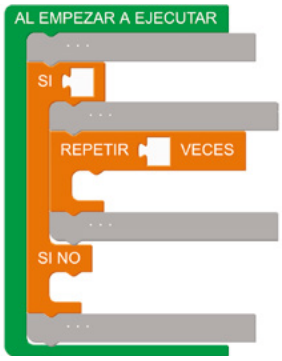
Actividad 3. Tarjetas de la parte 1

Tarjeta 1



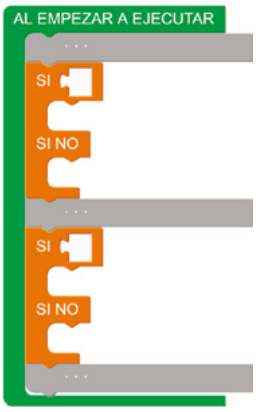
Una alternativa adentro de una repetición.

Tarjeta 2



Una repetición adentro de una alternativa.

Tarjeta 3



Una alternativa después de otra.

Actividad 3. Tarjetas de la parte 2

Tarjeta 1

Un procedimiento al principio, una alternativa con un procedimiento y un procedimiento al final.

**Pueden agregar algunas primitivas por fuera de los procedimientos si hace falta.*

**Pueden usar el mismo procedimiento más de una vez.*

Tarjeta 2

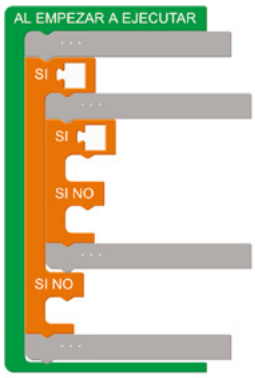
Primero, una alternativa con dos procedimientos. A continuación, otro procedimiento.

**Pueden agregar algunas primitivas por fuera de los procedimientos si hace falta.*

**Pueden usar el mismo procedimiento más de una vez.*

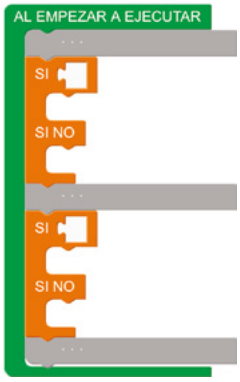
Actividad 3. Tarjetas de la parte 3

Tarjeta 1



Una alternativa condicional anidada dentro de otra.

Tarjeta 2



Una alternativa condicional a continuación de la otra.

Secuencias didácticas

> Unidad 3:

Interactividad y variables

12. **¿Podemos programar otros personajes?** Introducción a Scratch
13. **Programamos el personaje de un videojuego**. Interactividad y eventos
14. **Guardamos información**. Variables
15. **Programamos nuestro videojuego**. Variables, interactividad y un videojuego abierto

Secuencia didáctica 12

¿Podemos programar otros personajes?

Introducción a Scratch

¿Podemos elegir los personajes y los escenarios de nuestros proyectos? ¿Cómo construimos programas interactivos? ¿Qué tan diferentes son los entornos de programación Scratch y Pilas Bloques?

Esta secuencia consiste en una aproximación a Scratch como entorno didáctico de programación. La estrategia es presentar desafíos acotados que inviten a las y los estudiantes a explorar la nueva herramienta para conocer sus novedades (el trabajo con los personajes y su aspecto visual y la posibilidad de interactuar con el usuario) e identificar ideas y herramientas que ya conocen que pueden transferir a este nuevo entorno (como la repetición, los procedimientos o la alternativa condicional).

Actividad 1

Las y los estudiantes exploran proyectos existentes en la plataforma online de Scratch y abordan una serie de desafíos puntuales con el objetivo de realizar una exploración guiada de la herramienta. El objetivo es que conozcan los bloques de las categorías Movimiento, Apariencia y Eventos.

Actividad 2

Las y los estudiantes abordan un conjunto de desafíos que requieren aplicar en el nuevo entorno herramientas de programación vistas en secuencias anteriores: repetición, alternativa condicional y procedimientos.

Actividad 3

Se propone una instancia de trabajo libre en la que las y los estudiantes podrán programar un pequeño juego y personalizarlo como primer paso hacia el trabajo en entornos abiertos (a diferencia de los entornos orientados a desafíos). El objetivo es que construyan un proyecto que sientan propio y que, además, requiera del uso de algunas de las herramientas de programación conocidas.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: procedimientos, repeticiones, alternativa condicional, sensores, eventos.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad.

Objetivos de aprendizaje

- Conocer aspectos básicos y herramientas disponibles en el uso de Scratch.
- Transferir nociones de programación para resolver desafíos breves en un nuevo entorno.
- Introducir la noción de evento como mecanismo para elaborar programas interactivos.

Saberes previos de CC

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, procedimientos, repeticiones, alternativa condicional, sensores.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad, reutilización del código.

Materiales necesarios

- Dispositivos con Scratch 3 instalado o acceso a su versión en línea https://scratch.mit.edu.ar/

Actividad 1
Primeros pasos en Scratch

Se reparten tarjetas que invitan a las y los estudiantes a resolver desafíos puntuales y acotados en Scratch con el objetivo de que exploren el entorno y los modos de uso de esta herramienta, enfocándose especialmente en los bloques de las categorías Apariencia y Eventos.

Objetivo >

Se espera que las y los estudiantes se aproximen al entorno de Scratch y a la construcción de programas interactivos.

Inicio >

El **propósito de este momento** es motivar la introducción de Scratch como una herramienta para construir programas visuales e interactivos.

Orientaciones

Organizamos la clase en grupos heterogéneos pequeños[1]. Les pedimos que ingresen a Scratch[2]. A modo de motivación para conocer de qué se trata esta nueva herramienta de programación y qué tipo de programas nos permite construir, invitamos a las y los estudiantes a explorar la galería de proyectos de Scratch. Para acceder a la galería de proyectos, pueden ver los proyectos que se encuentran en la página principal como "Proyectos destacados", o bien, ingresar a la opción "***Explorar***" de la parte superior de la página principal.

Proponemos que recorran libremente la galería y observen en particular algún proyecto que les interese. Podrán ejecutar el programa presionando la bandera verde para interactuar como usuarios/as.

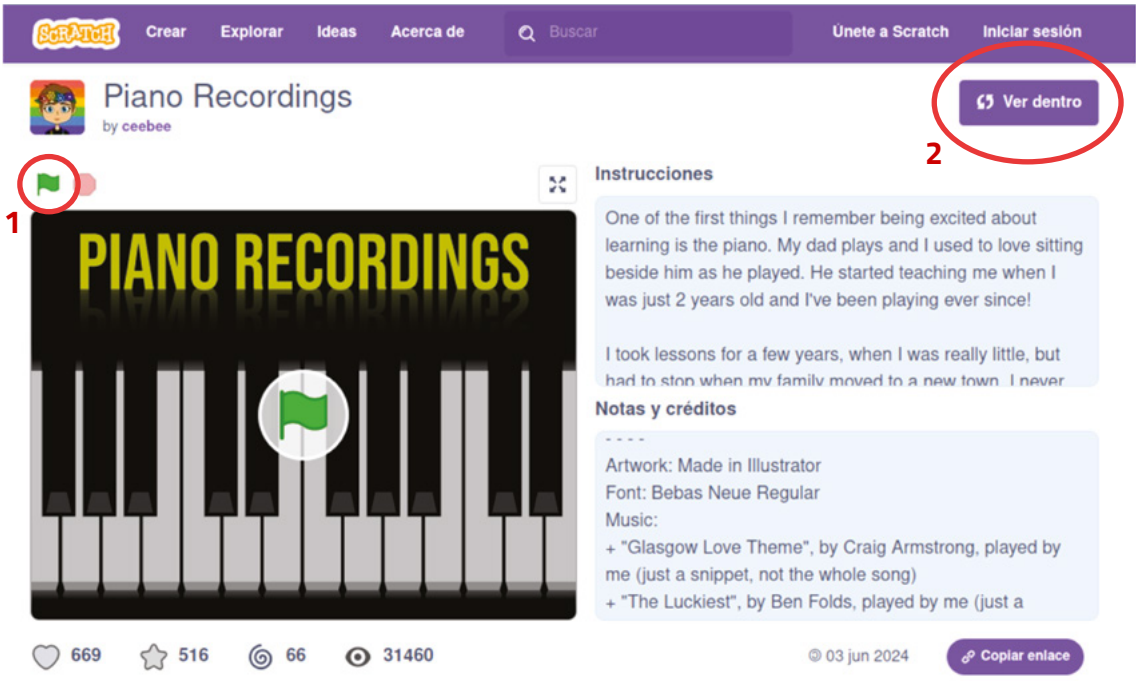
Al finalizar la exploración, invitamos a las y los estudiantes a compartir sus hallazgos sobre el entorno y describir el proyecto que eligieron explorar.

[1] Se pueden consultar dinámicas lúdicas para el armado de grupos heterogéneos en "Enseñar computación desde la mirada de la Educación Sexual Integral (ESI)", 10.

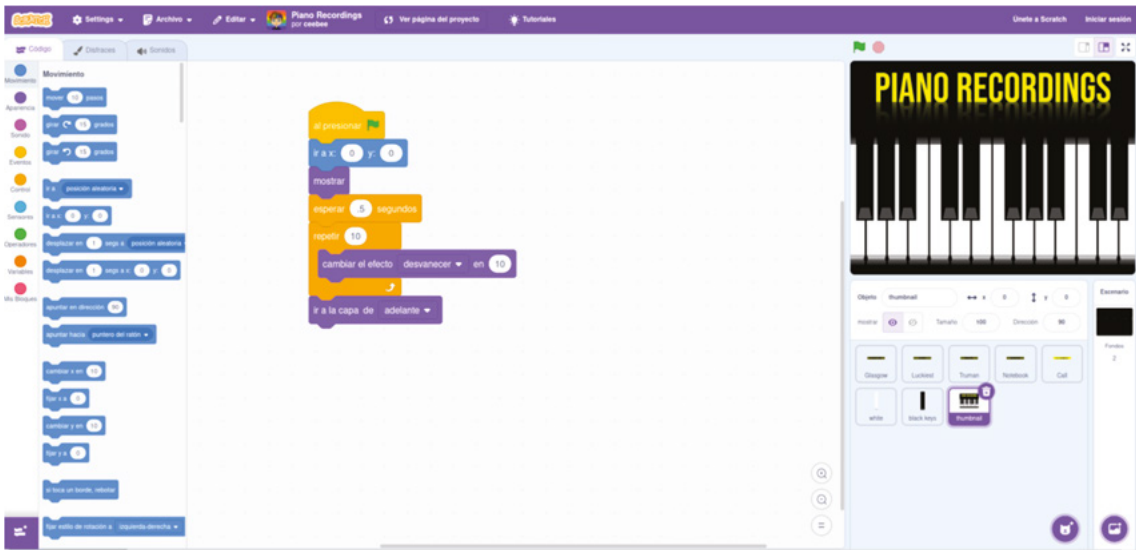
[2] Para usar Scratch pueden hacerlo con la versión sin conexión (previamente instalada, disponible en: https://scratch.mit.edu/download) o acceder a la versión en línea (https://scratch.mit.edu). En Linux también se puede usar el fork Scratux (https://github.com/scratux/scratux?tab=readme-ov-file).

¿Cuál es el objetivo o desafío del proyecto? ¿De qué manera se interactúa con él (por ejemplo, se mueve un personaje con el teclado, se presiona una tecla para avanzar, etc.)?

A continuación, les proponemos que presionen el botón "Ver Dentro", situado arriba y a la derecha de la página del proyecto, y vean cómo está programado el proyecto.



Página de un proyecto de Scratch. El botón de la bandera verde lo ejecuta (1) y el de "Ver dentro" muestra cómo está construido (2).



Vista de un proyecto de Scratch "por dentro".

Una vez que todo el grupo ha seleccionado y entrado en su proyecto, hacemos una puesta en común.

¿Cómo se construye un proyecto en Scratch? ¿Qué elementos identifican? ¿Encontraron similitudes con Pilas Bloques?

El objetivo de esta exploración es reconocer similitudes con PilasBloques y elementos que corresponden a conceptos que ya conocen. Por ejemplo, en ambos entornos, los programas se construyen con bloques y es posible identificar algunos comandos y anticipar qué realizan, como la alternativa condicional o la repetición. La novedad de Scratch es que en la construcción del programa se pueden elegir el escenario y los personajes.

Si queremos promover una exploración más profunda, podemos alentar a los grupos a que modifiquen el programa que analizaron a partir de tareas simples como cambiar un disfraz, alguna funcionalidad menor o algún comportamiento particular, que se desprenda de la exploración hecha hasta el momento.

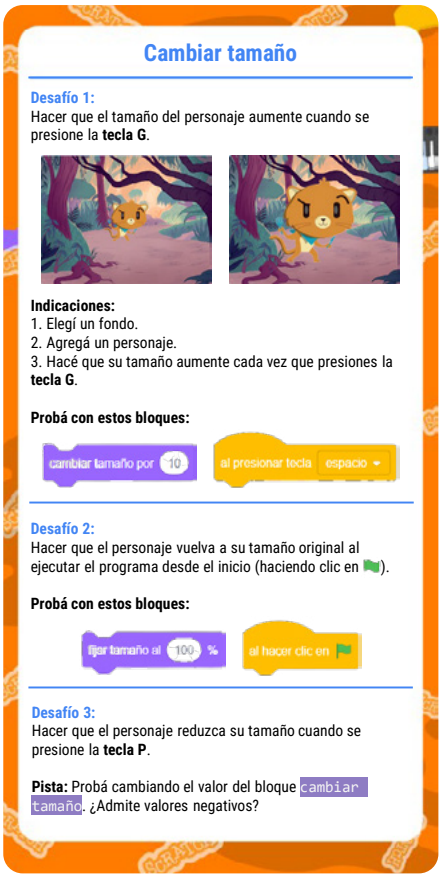
Al finalizar, las y los estudiantes podrán registrarse en Scratch para poder almacenar sus trabajos y compartirlos. Esto se hace desde el botón "Únete a Scratch" en ***https://scratch.mit.edu/join*** En caso de no poder hacer ese registro, las soluciones se pueden descargar en cada computadora. Independientemente de si trabajan en línea o no, es importante acordar una manera de guardar los trabajos.

Desarrollo >

El **propósito de este momento** es aproximarse al proceso de creación de un programa en *Scratch* desde cero a través de desafíos acotados.

Orientaciones

Entregamos a los grupos las tarjetas disponibles para esta actividad en el Anexo, formadas por tres desafíos. El recorrido propuesto guía la creación de un nuevo proyecto (elegir un escenario y un personaje), y explorar los bloques de eventos del teclado o del clic del mouse y otros de las categorías de Apariencia y Movimiento. Buscamos que atraviesen el proceso de crear un programa en Scratch desde cero sin perder el foco del objetivo de programación. Es probable que necesitemos asistir en el manejo del tiempo para que los grupos resuelvan rápidamente el aspecto visual y cuenten con el mayor tiempo posible para explorar y ensayar en el entorno la solución al desafío.



Cambiar tamaño

Desafío 1:
Hacer que el tamaño del personaje aumente cuando se presione la **tecla G**.

Indicaciones:
1. Elegí un fondo.
2. Agregá un personaje.
3. Hacé que su tamaño aumente cada vez que presiones la **tecla G**.

Probá con estos bloques:

Desafío 2:
Hacer que el personaje vuelva a su tamaño original al ejecutar el programa desde el inicio (haciendo clic en).

Probá con estos bloques:

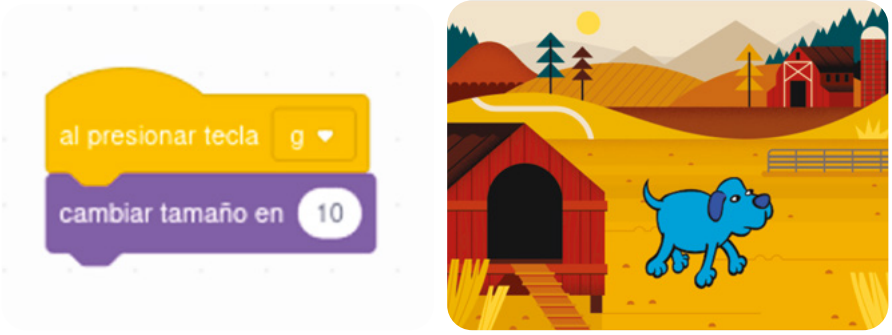
Desafío 3:
Hacer que el personaje reduzca su tamaño cuando se presione la **tecla P**.

Pista: Probá cambiando el valor del bloque `cambiar tamaño`. ¿Admite valores negativos?

Ejemplo de una de las tarjetas a entregar.

Brindamos un espacio para que los grupos avancen en la resolución del primer desafío con autonomía. Recorremos los puestos de trabajo y realizamos puestas en común cuando consideremos pertinente que compartan los hallazgos entre todos los grupos.

Cuando todos los grupos hayan resuelto el primer desafío, promovemos una puesta en común para que compartan cómo fue la manera en que llegaron a la solución del desafío y las categorías de bloques que exploraron.



Ejemplo de resolución del primer desafío de la tarjeta de ejemplo.

Luego de concluir la primera puesta en común, invitamos a las y los estudiantes a completar los siguientes desafíos. Es una oportunidad para

recordar la importancia de ir guardando los proyectos (ya sea en la cuenta en línea o en la computadora) para poder reutilizar (y resguardar) el trabajo previo.

Posible resolución de los tres desafíos de la tarjeta de ejemplo.

Cierre >

El **propósito de este momento** es explicitar las nociones de interactividad, evento y objeto en el trabajo con Scratch.

Orientaciones

Recuperamos el trabajo con las tarjetas y retomamos los intercambios orales sobre parecidos y diferencias entre Scratch y Pilas Bloques. A partir de lo que los grupos cuenten sobre los desafíos que resolvieron y cómo lo hicieron, nos interesa reforzar que:

- tanto en Pilas Bloques como en Scratch se utilizan bloques para construir el programa.
- en Scratch, los bloques son más variados y aparecen organizados en categorías, por ejemplo, las categorías Apariencia (que refiere a acciones que impactan en cómo se ven los personajes), Movimiento (que permite desplazar los personajes en el escenario) y Eventos (que define la interacción de la usuaria o el usuario con el programa a través del teclado o el mouse).

- Scratch permite crear **programas interactivos utilizando eventos**, es decir, programas que permiten considerar las acciones de quien está usando el programa (por ejemplo, si presiona una tecla o hace clic) durante la ejecución.
- los eventos son una herramienta que provee el entorno de Scratch para establecer qué debe suceder cuando el usuario interactúa con el programa, por ejemplo, son eventos los bloques Al presionar tecla [ ] o Al hacer clic en este objeto.
- Scratch nos permite agregar personajes u objetos al escenario y cada uno de ellos se programa por separado.

Actividad 2
Herramientas conocidas

A partir de un nuevo conjunto de tarjetas, las y los estudiantes resuelven desafíos que requieren recuperar las herramientas de programación que ya conocen (procedimientos, repetición simple y alternativa condicional).

Objetivos >

Se espera que las y los estudiantes:

- Empiecen a reconocer que la mayoría de los lenguajes de programación **tienen características en común** y la importancia de poder transferir lo que se sabe de un lenguaje a otro.
- Recuperen los conceptos de procedimiento, repetición simple y alternativa condicional para aplicarlos en la creación de programas interactivos.

Inicio >

El **propósito de este momento** es recuperar modos de trabajo en Scratch para desarrollar programas interactivos.

Conversamos con las y los estudiantes acerca de lo que experimentamos en la **Actividad 1** para recuperar la noción de programa interactivo, los bloques de evento y las nuevas categorías de bloques que permiten programar a los personajes (Movimiento, Apariencia, etc.).

Desarrollo >

El **propósito de este momento** es resolver desafíos que requieran transferir herramientas de programación conocidas al entorno Scratch.

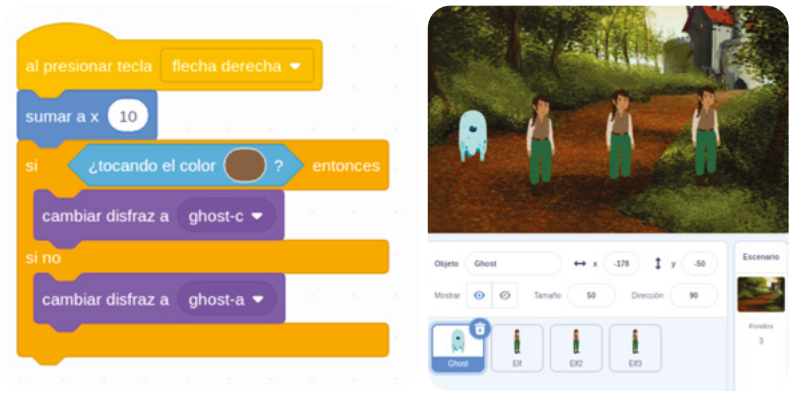
Presentamos un nuevo conjunto de tarjetas (ver "Tarjetas para la actividad 2", Anexo) y repartimos una a cada grupo.

Los desafíos de estas tarjetas requieren el uso de la alternativa condicional o de la repetición simple combinadas con la definición de procedimientos.

Una de las tarjetas a entregar.

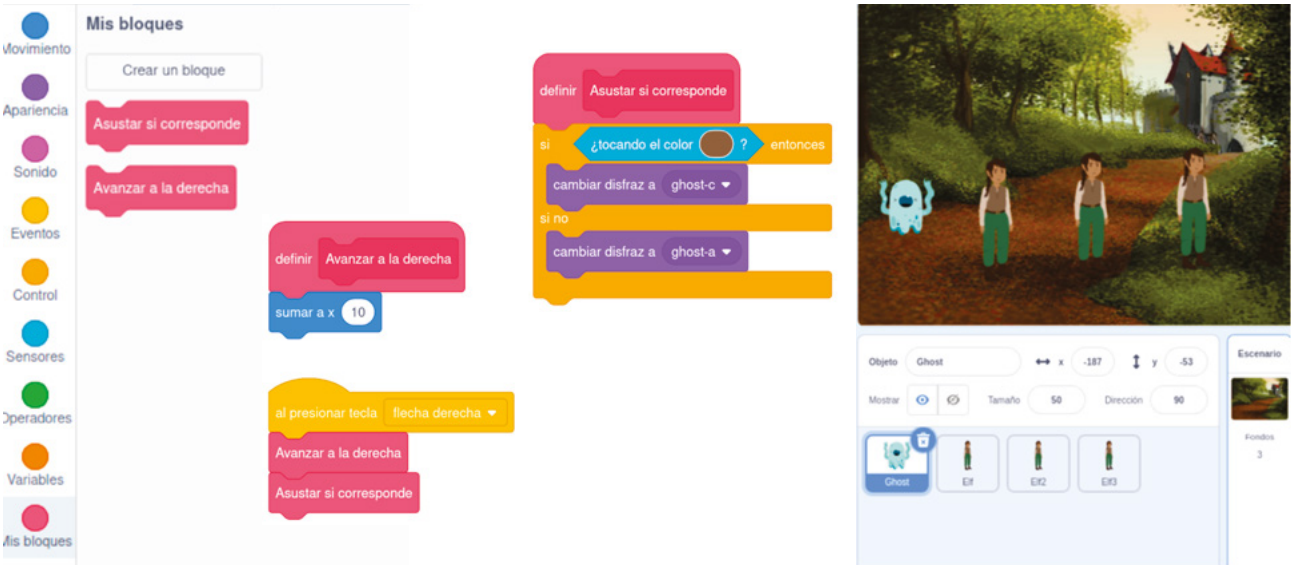
Proponemos a los grupos que comiencen a trabajar en la resolución de sus tarjetas y recorremos los puestos de trabajo mirando los progresos e interviniendo solo si fuera necesario.

En nuestras orientaciones reforzamos la importancia de la recuperación de los saberes previos y la identificación de similitudes entre Scratch y otros entornos conocidos para motivar y acompañar los procesos de transferencia. Nos interesa reforzar con estos desafíos que, a pesar de no haber trabajado ampliamente con Scratch, ya conocen muchas cosas sobre su modo de funcionamiento, pues es similar a otros entornos de programación. En este sentido, si bien es una de las primeras exploraciones, no están partiendo de cero.



Solución posible al desafío 1 de la tarjeta 5.

Una dificultad que puede presentarse al resolver el segundo desafío es la diferencia en las denominaciones entre Pilas Bloques y Scratch. Scratch denomina "Mis Bloques" a lo que en PilasBloques se llama "Procedimientos". Podemos relacionar la elección de este nombre con el hecho de que los procedimientos nos permiten definir bloques propios a partir de los bloques que provee el entorno o el desafío (por ejemplo, como vimos en la secuencia "Definimos nuestros bloques" de esta colección).



Solución al desafío 2 de la tarjeta 5.

Cierre >

El **propósito de este momento** es reconocer que, en este nuevo entorno, están presentes las mismas herramientas que abordaron en Pilas Bloques y explicitar los procesos de transferencia que hicieron al resolver los desafíos.

Armamos una puesta en común en la que cada grupo relata qué desafío tuvo que resolver, qué herramientas de programación requirió para eso, cómo las utilizó y cómo las utilizaban en otros entornos conocidos, como Pilas Bloques. Además de identificar las similitudes, es importante explicitar los procesos de transferencia que les permitieron usar las herramientas conocidas en un nuevo contexto.

Para concluir esta primera aproximación al nuevo entorno y reforzar la idea de que algunas de las herramientas conocidas están presentes en Scratch, enumeramos algunas e invitamos a categorizarlas según pertenezcan a Pilas Bloques (u otro entorno conocido previamente) o Scratch o ambos. Podemos realizarlo oralmente y anotar en una tabla en el piza-

rrón o distribuir papeles para que las y los estudiantes anoten los aspectos y los peguen en un afiche en el sector correspondiente.
Algunos aspectos posibles con los que trabajar son: Programación por bloques, Comandos básicos, Procedimientos, Repetición, Alternativa condicional, Sensores, Eventos, Escenario, Agregar/quitar objetos, Modificar tamaño, Mover personajes, Cambiar disfraces, Modificar escenario, Añadir pistas, Información de objetivo logrado.

Pilas Bloques	Scratch
Pistas Información de objetivo logrado	Programación por bloques Comandos básicos Procedimientos Repetición Alternativa condicional Sensores Escenario Mover personajes	Eventos Agregar/quitar objetos Modificar tamaño Disfraces Modificar/elegir escenario

Una clasificación posible de los aspectos de cada entorno de programación.

Durante la clasificación o con la tarea ya resuelta en el pizarrón, aprovechamos para señalar que:

- muchos aspectos fundamentales (las herramientas de programación aprendidas, la existencia del escenario, el uso de bloques, por ejemplo) pertenecen a ambos entornos; reconocer y aprovechar esas similitudes agiliza el dominio de un nuevo entorno.
- si algo no aparece exactamente cómo lo conocemos, no necesariamente significa que no exista o que no podamos aprovechar lo que ya sabemos en el nuevo entorno (por ejemplo, "Mis bloques" en Scratch cumple la misma función que los procedimientos en PilasBloques).
- los eventos son algo exclusivo de Scratch; Scratch nos habilita la posibilidad de crear programas interactivos, es decir, que contemplen las acciones de un/a usuario/a durante su funcionamiento.
- el escenario y los personajes en Pilas Bloques vienen dados por el desafío, mientras que en Scratch podemos elegirlos y modificarlos; utilizaremos Scratch para crear programas interactivos con los personajes y escenarios que queramos o necesitemos.

Actividad 3
Personalizamos un desafío

A modo de cierre de la secuencia, habilitamos un espacio para que las y los estudiantes continúen trabajando sobre los proyectos que empezaron, poniendo el foco en usar las herramientas que ya conocen para incorporar a sus programas un comportamiento dado (por ejemplo, que el personaje regrese al borde al tocar un objeto). Guiamos una instancia de trabajo más libre en la que puedan aprovechar las posibilidades que brinda un entorno abierto para personalizar (y apropiarse de) los proyectos que están construyendo.

Objetivos >

Se espera que las y los estudiantes:

- Recuperen los conceptos de procedimiento, repetición simple, alternativa condicional, sensores y/o eventos para construir comportamientos preestablecidos.
- Se familiaricen con la creación de programas interactivos.

Inicio >

El **propósito de este momento** es reforzar la idea de que existen similitudes entre los entornos o los lenguajes de programación y, por lo tanto, se puede aprovechar lo que sabe de unos en otros.

Orientaciones

Comenzamos con una breve puesta en común para recuperar las similitudes que encontraron con Pilas Bloques y los hallazgos en Scratch.

Invitamos a los grupos a que recuperen alguno de los programas que construyeron en las actividades anteriores de la secuencia para aumentarlo según sus preferencias en línea con algunas consignas que les brindaremos.

Reforzamos que, en esta actividad, podrán aprovechar las posibilidades que brinda un entorno abierto como Scratch para personalizar e incorporar sus preferencias al proyecto.

Desarrollo >

El **propósito de este momento** es que las y los estudiantes realicen una primera experiencia de trabajo en un entorno abierto en el que se combine el cumplimiento de un objetivo de programación con la posibilidad de personalización del programa.

Orientaciones

Se trabajará con una consigna común y luego con otras que pueden variarse de acuerdo con el tiempo disponible, cantidad de estudiantes, grado de comprensión, etc.

El desarrollo de esta actividad se hará en dos partes. La primera parte atenderá **una consigna común** a todos los grupos: "Permitir que el usuario controle el movimiento de un personaje, por ejemplo, hacer que se mueva con las flechas del teclado".

Para la segunda parte, podemos asignar las tarjetas con las consignas o dejar que los grupos las elijan (ver "Tarjetas para la actividad 3", Anexo). Estas consignas se refieren a comportamientos que debe tener el programa (cómo debe moverse un personaje, cuándo debe cambiar el escenario, etc.), pero no dicen nada sobre el aspecto del programa (qué personaje, qué escenario, etcétera).

Cada consigna por separado debe apuntar a obtener una nueva versión del programa con una función añadida. Buscaremos lograr que obtengan más de una versión en vistas de construir programas cada vez más complejos, como pequeños videojuegos.

La idea es que puedan cumplir las consignas apelando a las herramientas de programación que ya conocen. Es importante que registren, para compartir luego con los demás grupos, qué parte del programa resuelve cada consigna, qué bloques eligieron para eso y por qué. Pueden sistematizarlo durante la resolución del desafío o recuperarlo con el programa terminado.

Nuestras orientaciones deben apuntar a agilizar la etapa de la personalización (elección de los fondos, los personajes, los disfraces, etc.) para que se concentren en la programación. Para resolver la etapa de programación, apelamos a experiencias previas, tanto durante esta secuencia como durante el trabajo previo en otros entornos. La dinámica de trabajo dependerá mucho de la familiaridad de los grupos con la herramienta y la facilidad con la que identifiquen los bloques que necesitan para cumplir con cada consigna. Es importante que brindemos acompañamiento, pero teniendo presente que el objetivo de esta instancia es que puedan experimentar un mayor grado de autonomía. Si detectamos dificultades comunes, podemos realizar interrupciones para resolverlas entre todas y todos.

Cierre >

El **propósito de este momento** es reforzar la idea de entorno abierto, reconocer herramientas de programación comunes y explicitar acciones de transferencia de saberes entre los distintos entornos de programación.

Promovemos un intercambio entre los grupos para que compartan sus producciones. Podemos proyectar los programas para que los vea todo el curso o habilitar la circulación entre los grupos a modo de feria y que cada grupo presente su programa.

¿Qué consignas resolvieron? ¿En qué orden las abordaron? ¿Qué bloques usaron para resolver cada una? ¿Podemos identificar qué parte del programa resuelve cada consigna? ¿Por qué?

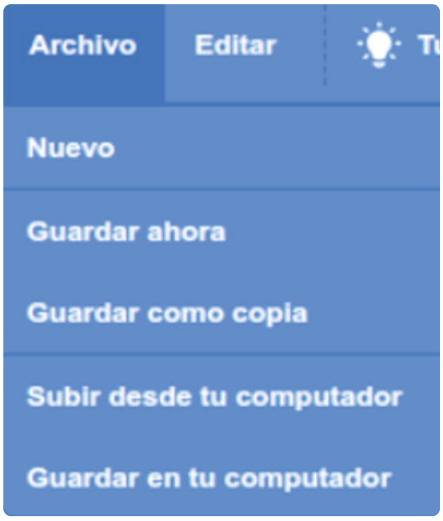
En el diálogo entre los grupos, es importante que identifiquen en el proceso de construcción cuándo recurrieron a saberes previos y de qué manera los adaptaron (es decir, que expliciten el **proceso de transferencia**). Por ejemplo, "Para conseguir que un personaje cambie de aspecto cuando pasa sobre otro, colocamos una alternativa condicional a continuación del movimiento. Pensamos en esta solución porque había dos casos posibles (que estuviera o no estuviera el objeto) al igual que en los desafíos de Pilas Bloques cuando podíamos encontrar o no una lata para recoger. Además, igual que en Pilas Bloques, usamos un sensor para construir la condición y colocamos los bloques correspondientes a cada caso dentro de cada rama de la alternativa".

Abstraer el proceso de construcción y explicitar los razonamientos probablemente requiera de una moderación intensa del relato. Sin embargo, **es sumamente valioso explicitar estas acciones de transferencia**, ya que, de otra manera, pueden pasar desapercibidas y no formar parte de los aprendizajes logrados durante la secuencia.

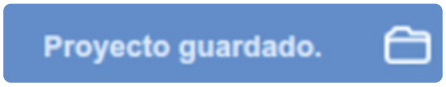
Anexo

Proceso de registro en Scratch

1. Una vez que presionamos "Únete a Scratch", comenzaremos el proceso de registro en esta plataforma.

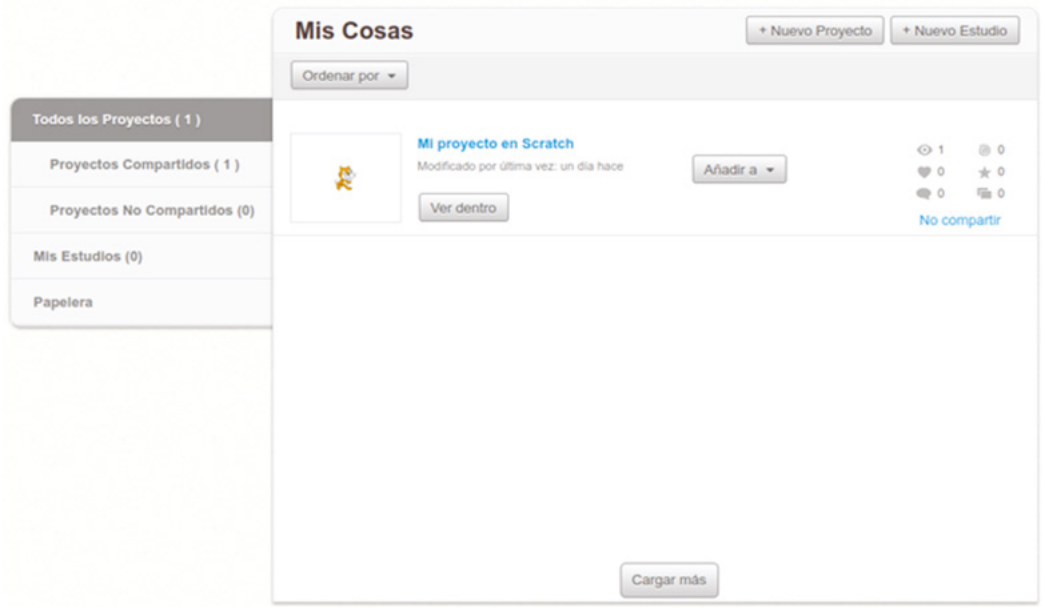


2. Allí se deberán ingresar ciertos datos personales como usuario, contraseña, país, correo electrónico, etc. El correo servirá para confirmar estos datos y dar el alta a la cuenta. Luego, usando el botón "Guardar Ahora" podrán almacenar sus trabajos y compartirlos con el resto.



3. Cuando el proyecto esté guardado podremos ver la confirmación

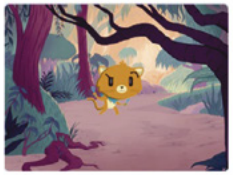
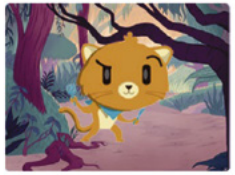
4. Y en "Mis cosas" estarán los trabajos que hayan guardado con esa modalidad.



Tarjetas para la Actividad 1

Cambiar tamaño

Desafío 1:
Hacer que el tamaño del personaje aumente cuando se presione la **tecla G**.

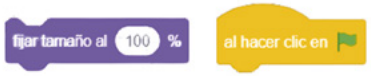



Indicaciones:
1. Elegí un fondo.
2. Agregá un personaje.
3. Hacé que su tamaño aumente cada vez que presiones la **tecla G**.

Probá con estos bloques:

Desafío 2:
Hacer que el personaje vuelva a su tamaño original al ejecutar el programa desde el inicio (haciendo clic en ⚑).

Probá con estos bloques:



Desafío 3:
Hacer que el personaje reduzca su tamaño cuando se presione la **tecla P**.

Pista: Probá cambiando el valor del bloque `cambiar tamaño`. ¿Admite valores negativos?

Aplicar efecto visual

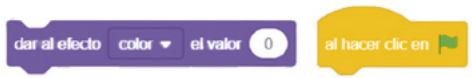
Desafío 1:
Hacer que el tamaño del personaje se pixele cuando se presione la **tecla P**.


Indicaciones:
1. Elegí un fondo.
2. Agregá un personaje.
3. Hacé que el personaje se pixele más y más cada vez que presiones la **tecla P**.

Probá con estos bloques:

Desafío 2:
Hacer que el personaje recupere su aspecto original al ejecutar el programa desde el inicio (haciendo clic en ⚑).

Probá con estos bloques:



Desafío 3:
Hacer que el personaje revierta el pixelado cuando se presione la **tecla D**.

Pista: Probá cambiando el valor del bloque `sumar al efecto`. ¿Admite valores negativos?

Girar

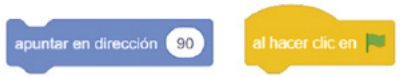
Desafío 1:
Hacer que el personaje gire en sentido horario cuando se presione la **tecla →**.


Indicaciones:
1. Elegí un fondo.
2. Agregá un personaje.
3. Hacé que el personaje gire en sentido horario cada vez que presiones la **tecla →**.

Probá con estos bloques:

Desafío 2:
Hacer que el personaje vuelva a su orientación original al ejecutar el programa desde el inicio (haciendo clic en ⚑).

Probá con estos bloques:



Desafío 3:
Hacer que el personaje gire en sentido antihorario cuando se presione la **tecla ←**.

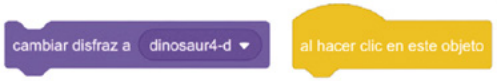
Pista: Probá con el bloque 

Cambiar disfraz y hablar

Desafío 1:
Hacer que el disfraz del personaje cambie cuando se hace **clic** sobre él.

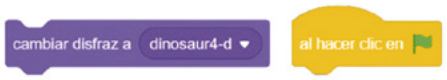

Indicaciones:
1. Elegí un fondo.
2. Agregá un personaje.
3. Hacé que su disfraz cambie a otro cuando hagas **clic** sobre él.

Probá con estos bloques:

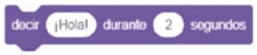


Desafío 2:
Hacer que el personaje vuelva al disfraz original al ejecutar el programa desde el inicio (haciendo clic en ⚑).

Probá con estos bloques:



Desafío 3:
Hacer que el personaje, además de cambiar de disfraz, **diga algo**, como una caricatura.

Pista: Probá con el bloque 

Mover aleatoriamente e ir a posición

Desafío 1:
Hacer que el personaje vaya a posición aleatoria en el escenario cuando se hace **clic** sobre él.


Indicaciones:
1. Elegí un fondo.
2. Agregá un personaje.
3. Hacé que el personaje se mueva a una posición aleatoria en el escenario cada vez que hagas **clic** sobre él.

Probá con estos bloques:

Desafío 2:
Hacer que el personaje aparezca en el centro del escenario al ejecutar el programa desde el inicio (haciendo clic en ⚑).

Probá con estos bloques:

Desafío 3:
Hacer que el personaje aparezca en una posición aleatoria antes de moverse.

Pista: Probá con los bloques 

Mover horizontalmente

Desafío 1:
Hacer que el personaje se mueva a la derecha cuando se presione la **tecla** →.


Indicaciones:
1. Elegí un fondo.
2. Agregá un personaje.
3. Hacé que el personaje se mueva a la derecha cada vez que presiones la **tecla** →.

Probá con estos bloques:

Desafío 2:
Hacer que el personaje aparezca en el centro horizontal al ejecutar el programa desde el inicio (haciendo clic en ⚑).

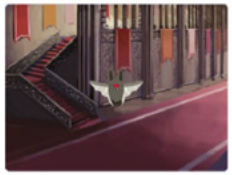
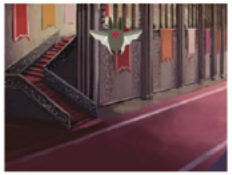
Probá con estos bloques:

Desafío 3:
Hacer que el personaje se mueva a la izquierda cuando se presione la **tecla** ←.

Pista: Probá cambiando el valor del bloque `sumar a x`. ¿Admite valores negativos?

Mover verticalmente

Desafío 1:
Hacer que el personaje se mueva hacia arriba cuando se presione la **tecla ↑**.

Indicaciones:
1. Elegí un fondo.
2. Agregá un personaje.
3. Hacé que el personaje se mueva hacia arriba cada vez que presiones la **tecla ↑**.

Probá con estos bloques:

Desafío 2:
Hacer que el personaje aparezca en el centro vertical al ejecutar el programa desde el inicio (haciendo clic en ⚑).

Probá con estos bloques:

Desafío 3:
Hacer que el personaje se mueva hacia abajo cuando se presione la **tecla ↓**.

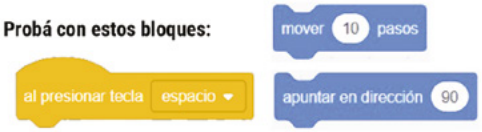
Pista: Probá cambiando el valor del bloque `sumar a y`. ¿Admite valores negativos?

Mover en una dirección

Desafío 1:
Hacer que el personaje se mueva en diagonal cuando se presione la **tecla I**.


Indicaciones:
1. Elegí un fondo.
2. Agregá un personaje.
3. Hacé que el personaje se mueva en diagonal cada vez que presiones la **tecla I**.

Probá con estos bloques:



Desafío 2:
Hacer que el personaje aparezca en el centro del escenario al ejecutar el programa desde el inicio (haciendo clic en ⚑).

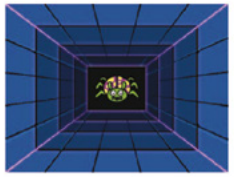
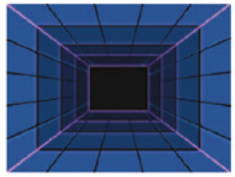
Probá con estos bloques:

Desafío 3:
Hacer que el personaje en la dirección contraria a la original cuando se presione la **tecla V**.

Pista: Probá cambiando el valor del bloque `apuntar en dirección`.

Mostrar y esconder

Desafío 1:
Hacer que el personaje desaparezca cuando se hace **clic** sobre él.

Indicaciones:
1. Elegí un fondo.
2. Agregá un personaje.
3. Hacé que el personaje desaparezca cuando hagas **clic** sobre él.

Probá con estos bloques:

Desafío 2:
Hacer que el personaje vuelva a aparecer en el escenario al ejecutar el programa desde el inicio (haciendo clic en ⚑).

Probá con estos bloques:

Desafío 3:
Hacer que el personaje piense algo antes de desaparecer.

Pista: Probá con el bloque

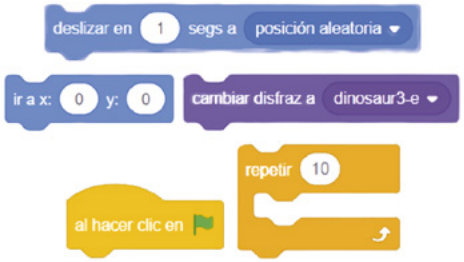
Tarjetas para la Actividad 2

1

Desafío 1:
Hacer que el pterodáctilo haga un berrinche y, luego, se calme.

Indicaciones:
1. Elegí un escenario.
2. Agregá un pterodáctilo (*Dinosaur3*).
3. Hacé que se enoje (*disfraz dinosaur3-d*) y vuele a 10 posiciones aleatorias al hacer clic sobre .
4. Hacé que luego se calme (*disfraz dinosaur3-e*) en el centro del escenario.

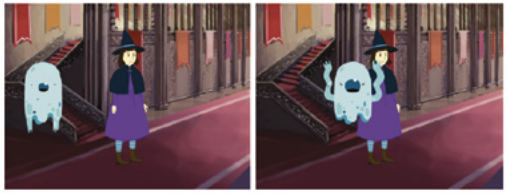
Probá con estos bloques:



Desafío 2:
Crear y utilizar los bloques:

2

Desafío 1:
Hacer que el fantasma intente asustar a la bruja cuando pase sobre ella.



Indicaciones:
1. Elegí un escenario.
2. Agregá un fantasma (*Ghost*) y una bruja (*Witch*).
3. Hacé que el fantasma se mueva a la derecha al presionar la tecla →.
4. Hacé que asuste (*disfraz ghost-c*) si está sobre la bruja o vuelva a la normalidad (*disfraz ghost-a*).

Probá con estos bloques:

Desafío 2:
Crear y utilizar los bloques:

3

Desafío 1:
Hacer que el fantasma intente asustar a la bruja cuando pase sobre ella.

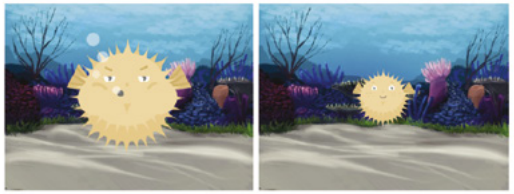
Indicaciones:
1. Elegí un escenario.
2. Agregá un mago (*Wizard*) y una varita (*Wand*).
3. Hacé que el mago se mueva a la derecha al presionar la tecla →.
4. Hacé que diga un hechizo, por ejemplo "¡Avada kadabra!", si está sobre la varita mágica.

Probá con estos bloques:

Desafío 2:
Crear y utilizar los bloques:

4

Desafío 1:
Hacer que el pez globo se enoje y luego se tranquilice.



Indicaciones:
1. Elegí un escenario.
2. Agregá un pez globo (*Pufferfish*).
3. Hacé que se enoje (*disfraz pufferfish-c*) y aumente su tamaño 10 veces al hacer clic en .
4. Hacé que luego reduzca su tamaño 10 veces y se tranquilice (disfraz *pufferfish-a*).

Probá con estos bloques:

Desafío 2:
Crear y utilizar los bloques:

5

Desafío 1:
Hacer que el fantasma intente asustar a las elfas cuando pase sobre ellas.

Indicaciones:
1. Elegí un escenario.
2. Agregá un fantasma (*Ghost*) y tres elfas (*Elf*).
3. Hacé que el fantasma se mueva a la derecha al presionar la tecla →.
4. Hacé que asuste (disfraz *ghost-c*) si está sobre una elfa o vuelva a la normalidad (disfraz *ghost-a*).

Probá con estos bloques:

Desafío 2:
Crear y utilizar los bloques:

6

Desafío 1:
Hacer que la bruja lance un hechizo cuando pase sobre una varita mágica.

Indicaciones:
1. Elegí un escenario.
2. Agregá una bruja (*Witch*) y tres varitas (*Wand*).
3. Hacé que la bruja se mueva a la derecha al presionar la tecla →.
4. Hacé que diga un hechizo, por ejemplo "¡Expecto patronum!", si está sobre una varita mágica.

Probá con estos bloques:

Desafío 2:
Crear y utilizar los bloques:

Tarjetas para la Actividad 3

Ubicaciones al azar

Hacer que al comenzar el programa alguno/s de los objetos o personajes se ubiquen aleatoriamente en el escenario.

Pista: Explorar la categoría Eventos y Movimiento.

Se encuentran dos personajes

Cambie el fondo y/o suene un sonido.

Pista: Definan procedimientos para no reutilizar partes del programa.

Un obstáculo

Si el personaje se encuentra con un obstáculo, debe volver a la esquina superior izquierda. Para el obstáculo, pueden agregar un objeto nuevo o usar uno que ya esté en el programa.

Pista: Definir un procedimiento o bloque que se llame “Volver si chocó”.

Alcanzar un objetivo

Hacer que si un personaje alcanza un objeto u otro personaje cambie de disfraz, indicando que cumplió su objetivo.

Pista: Pueden definir procedimientos para no repetir partes del programa.

Una pared

Trazar una línea en el escenario (una pared) y hacer que, si un personaje la toca, vuelva a su posición inicial.

Pista: Dibujen la línea de un color que no aparezca en ningún otro lugar del escenario.

Movimiento permanente

Algún personaje u objeto (agregado o existente) debe moverse solo durante toda la ejecución del programa.

Pista: Revisar los bloques de la categoría Control.

“Objetivo: llegar arriba”

Hacer que cuando un personaje toca el borde superior cambie de disfraz, indicando que cumplió con su objetivo.

Pista: Exploren los sensores. ¿Pueden saber qué borde está tocando un objeto? ¿Qué movimientos pueden hacer que un objeto toque el borde superior?.

Dar la vuelta

Hacer que cuando un personaje toca el borde derecho, aparezca por el borde izquierdo.

Pista: Exploren las coordenadas y los bloques asociados a ellas.
Pista: Exploren los sensores. ¿Pueden saber qué borde está tocando un objeto? ¿Qué movimientos pueden hacer que un objeto toque el borde derecho?

Controlar otro personaje

Permitir que el usuario mueva a otro objeto o personaje.

Pista: Puede usar otras teclas (por ejemplo) A, S, D, W.

Soluciones posibles para cada tarjeta de la Actividad 3

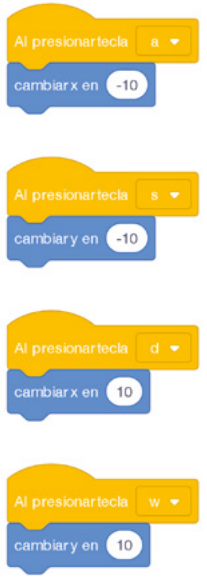
Un obstáculo

Definimos un procedimiento con una alternativa condicional para que, cada vez que se mueve el personaje, evaluemos si chocó con el obstáculo y vuelva a su lugar.

Movimiento permanente

Usamos el bloque "Por siempre" con bloques de movimiento adentro.

Controlar otro personaje



Definimos nuevos eventos del teclado en otro objeto del videojuego.

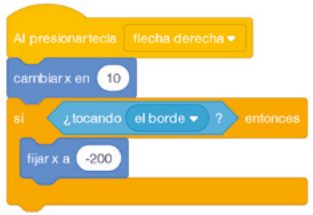
Se encuentran dos personajes

Definimos un procedimiento con una alternativa condicional para evaluar si uno de los personajes chocó con el otro y lo usamos en sus 4 eventos de movimiento.

Una pared

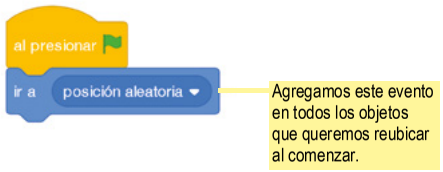
Dibujamos una línea fucsia para asegurarnos de que fuera lo único fucsia en el escenario. Luego, definimos un procedimiento con una alternativa condicional para detectar si el personaje se había movido sobre la línea usando el sensor ¿Tocando color?

Dar la vuelta



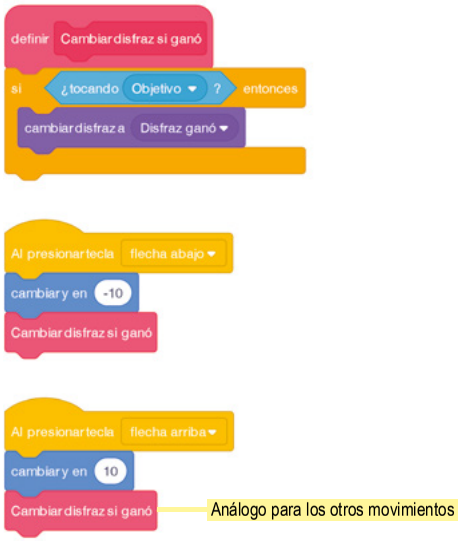
Cambiamos solo la coordenada X del personaje para desplazarlo al otro lado del escenario. Usamos una alternativa condicional con el sensor ¿Tocando borde? solo en el evento de moverse a la derecha.

Ubicaciones al azar



Usamos el evento "Al presionar bandera verde" y el bloque de movimiento "Ir a posición aleatoria".

Alcanzar un objetivo



Definimos un procedimiento que evalúa si el personaje tocó al objetivo con un sensor y una alternativa condicional. Lo utilizamos en los cuatro eventos de movimiento del personaje.

Objetivo: llegar a arriba

Usamos una alternativa condicional con el sensor ¿tocando borde?; lo colocamos solo en el evento de mover hacia arriba al personaje para identificar solo cuando alcanza el borde de arriba.

Secuencia didáctica 13

Programamos el personaje de un videojuego

Interactividad y eventos

¿Qué es un proyecto de programación? ¿Podemos construir un videojuego sencillo en Scratch?

En esta secuencia se desarrolla un proyecto que consiste en programar un pequeño videojuego usando Scratch para recuperar las herramientas de programación vistas en secuencias anteriores. El juego se elabora en etapas, incorporando dificultades de manera progresiva.

Actividad 1

Las y los estudiantes construyen una solución para mover un personaje con las flechas del teclado.

Actividad 2

Programan las colisiones con los obstáculos para que estos impidan el avance del personaje.

Actividad 3

En esta última parte del proyecto, agregan el comportamiento de los enemigos y del objetivo final. También, pueden dedicarse a personalizar su videojuego.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, procedimientos, repeticiones, alternativa condicional, sensores, eventos.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad.

Objetivos de aprendizaje

- Recuperar herramientas de programación conocidas (comandos, procedimientos, alternativa condicional, eventos) para resolver problemas asociados al desarrollo de videojuegos.
- Identificar las colisiones como un problema frecuente en el desarrollo de videojuegos y elaborar estrategias para resolverlo.
- Transitar la elaboración de un proyecto de programación en un proceso de desarrollo incremental.

Saberes previos de CC

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, procedimientos, repeticiones, alternativa condicional, sensores, eventos.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad.

Materiales necesarios

- Dispositivos con Scratch 3 instalado o acceso a su versión en línea https://scratch.mit.edu.ar/
- Proyecto de Scratch con el videojuego terminado (Nacles y la manzana polar_completo.sb3).
- Proyecto de Scratch para completar (Nacles y la manzana polar.sb3).

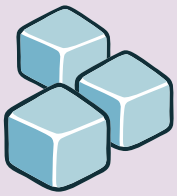
Actividad 1
Mover un personaje

Las y los estudiantes comienzan el desarrollo del videojuego a partir de una versión preexistente que modificarán en sucesivas etapas.

Objetivo >

Se espera que las y los estudiantes:

- Se aproximen a las nociones de desarrollo incremental y proyecto de programación, en un recorrido gradual de actividades orientadas a un producto final concreto.
- Ejerciten la creación de programas interactivos en Scratch utilizando eventos para mover un personaje.



Inicio >

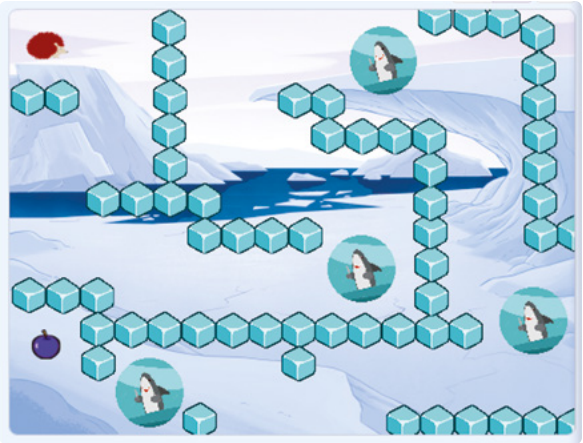
El **propósito de este momento** es presentar el proyecto de programación sobre el que trabajaremos en la secuencia y distribuir los materiales necesarios para comenzar.

Orientaciones

Comenzamos la actividad presentando el trabajo de la secuencia como un primer proyecto de programación: a partir de sucesivas intervenciones se irá construyendo gradualmente un videojuego, es decir, será un **proceso incremental**.

Para explicar cómo funciona el videojuego a construir y motivar el proyecto, podemos proyectar la versión final y proponerles jugarla algunas veces desde nuestra computadora. Encontrarán una versión completa del juego en el archivo Nacles y la manzana polar_completo.sb3 que acompaña esta secuencia.

Al jugarlo, verán que el objetivo es lograr que el personaje (Nacles) llegue al final de un laberinto para comer una manzana. El obstáculo a sortear son unos tiburones que, en caso de que el personaje los toque, lo envían al principio del recorrido.



El escenario del videojuego.

En Scratch no es posible distribuir un proyecto restringiendo el acceso a los bloques con los que está programado. Por este motivo, recomendamos que el proyecto completo no se distribuya a las computadoras de las y los estudiantes.

Material para docentes

Compartir un proyecto de Scratch

Para subir un proyecto de Scratch a nuestra cuenta, debemos iniciar sesión con nuestro usuario, crear un proyecto nuevo y en la pestaña "Archivo" elegir "Subir un archivo desde tu computador" o "Cargar un archivo desde tu ordenador" (según la traducción elegida). Luego, para compartirlo, podemos hacerlo desde la página del proyecto o desde el editor de bloques, cliqueando el botón "Compartir" de la barra superior.

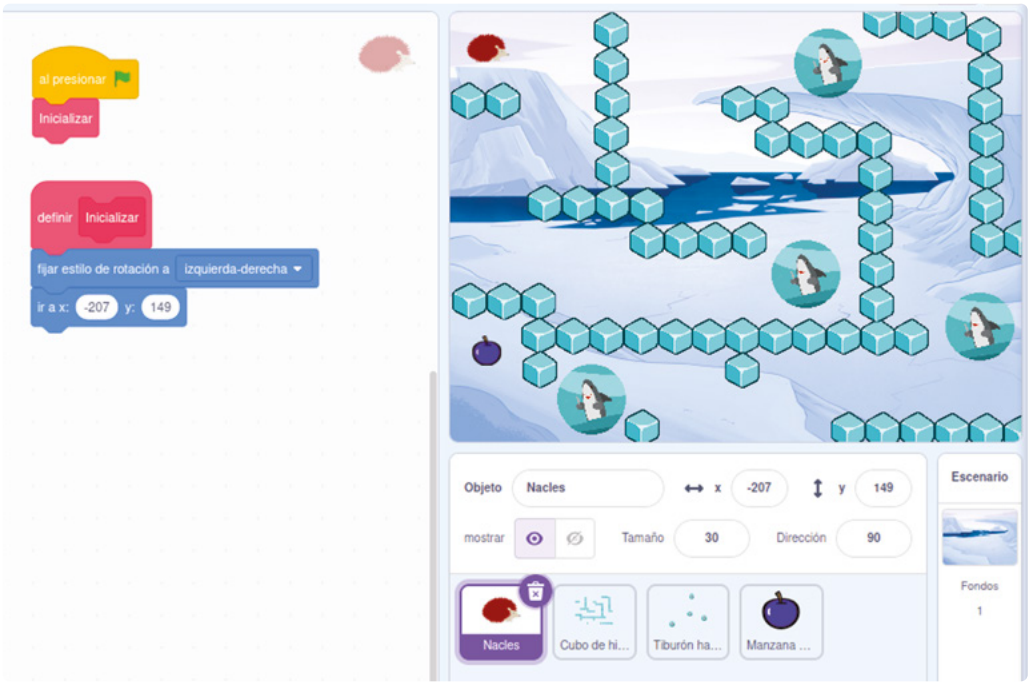
La versión en línea de Scratch permite "Reinventar" un proyecto: crear una copia del proyecto (que estará asociada al proyecto original) para modificarlo, aumentarlo, personalizarlo, etcétera. De esta forma, las y los estudiantes pueden experimentar una práctica frecuente en el ámbito del desarrollo de software que puede potenciar un proceso incremental: la **construcción colaborativa**. Es importante tener en cuenta que todos los proyectos que se creen como reinvenciones aparecerán enlazados en la página del proyecto original. Esto permite que cualquier persona que ingrese a la página del proyecto pueda acceder a estos proyectos y a cómo están programados en su versión final.

Terminada la exploración del proyecto completo, proponemos a las y los estudiantes que formen grupos heterogéneos[1] para trabajar con una versión del programa en sus computadoras. Para comenzar a programar, distribuimos el proyecto (Nacles y la manzana polar.sb3) que solo contiene la disposición del laberinto y los objetos para que las y los estudiantes avancen sobre la programación del comportamiento de los personajes y los obstáculos. Para ello, podemos distribuir copias del archivo o subirlo a nuestro usuario de Scratch y compartirlo para que los grupos lo "reinventen".

A partir de esta exploración, también, compartimos las sucesivas etapas en las que desarrollaremos el juego:

1. Programaremos al personaje para que pueda moverse con las flechas del teclado.
2. Programaremos que los cubos de hielo y los borden del escenario impidan el avance del personaje.
3. Agregaremos el comportamiento del objetivo (la manzana) y los enemigos (los tiburones).

Será importante tener presente este recorrido como referencia durante el desarrollo de la secuencia.



Aspecto del programa incompleto.

[1] Se pueden consultar dinámicas lúdicas para el armado de grupos heterogéneos en "Enseñar computación desde la mirada de la Educación Sexual Integral (ESI)", 10.

Desarrollo >

El **propósito de este momento** es recuperar la noción de evento para construir programas interactivos.

Orientaciones

Retomando la idea de desarrollo incremental, asociada a la noción de proyecto, presentamos la primera consigna: **lograr que el personaje se mueva en las cuatro direcciones usando las flechas del teclado**. Es importante tener presente que se trata de un primer paso y, por lo tanto, no es necesario preocuparse por otros.

Destinamos un momento a analizar algunas cuestiones puntuales que emergen de la consigna antes de que comiencen a programar. Podemos orientar el proceso con preguntas que ayuden a pensar la estrategia.

> *Para lograr resolver la consigna, ¿qué herramientas de programación usarán? ¿Qué categorías de bloques se necesitan? ¿Cómo determina Scratch la dirección en la que se mueve un objeto en el escenario?*

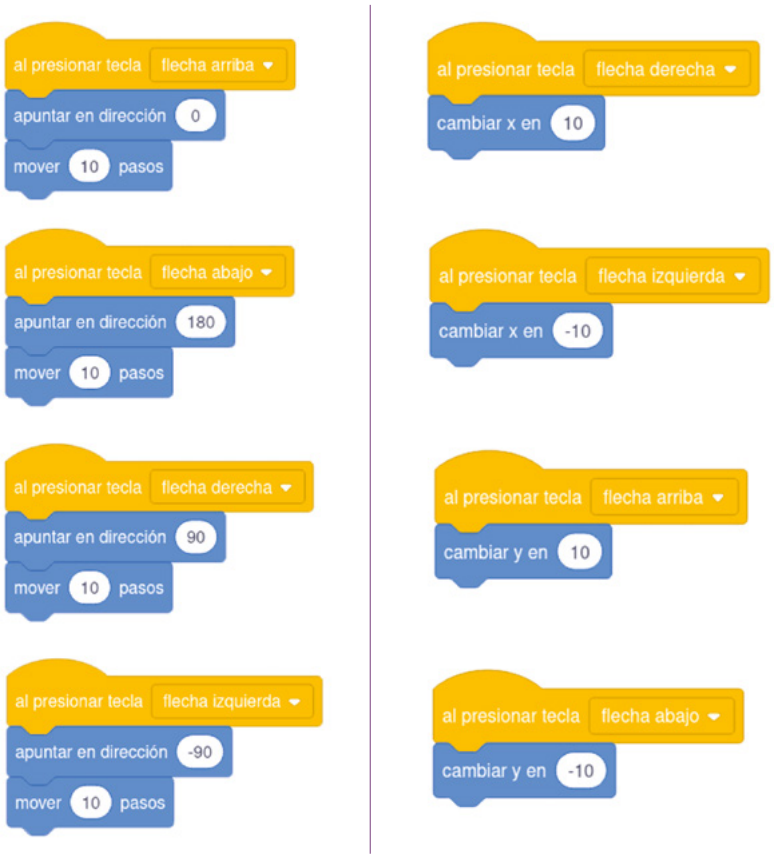
Estas preguntas intentan recuperar la noción de evento asociada a la interactividad, en particular, los eventos asociados al teclado. Estos bloques aparecen en la categoría Eventos. También, llamar la atención sobre la categoría Movimiento para resolver el problema de mover al personaje.

En cuanto a la dirección del movimiento del personaje, pueden utilizarse los bloques cambiar x en (■) y cambiar y en (■). También se puede resolver con los bloques apuntar en dirección (■) o girar (■) grados y mover (■) pasos. La solución elegida dependerá de los saberes previos de los estudiantes. Si necesitamos reforzar alguna explicación, será importante hacerlo con un nivel de profundidad acorde a la tarea que estamos resolviendo, es decir, no desarrollar completamente el sistema de coordenadas y ángulos de Scratch, sino lo mínimo indispensable para completar esta tarea.

Utilizar el bloque apuntar en dirección (■) presenta la ventaja de que la dirección se puede ajustar gráficamente con una rueda de 360°. Esto habitualmente facilita la comprensión del bloque, porque resulta más concreto que los demás bloques.

El selector gráfico para ajustar la dirección en el bloque apuntar en dirección (■).

Como vimos en la planificación de las etapas, no importa que el personaje avance por el laberinto o esquive los tiburones; esas cuestiones serán algunas de las tareas a resolver en próximas etapas.



Dos maneras de resolver el movimiento del personaje: a la izquierda, utilizando los bloques apuntar en dirección (■) y mover (■) pasos ; a la derecha, modificando directamente las coordenadas.

Cierre >

El **propósito del cierre** es socializar las estrategias de cada grupo.

Orientaciones

Invitamos a las y los estudiantes a que compartan brevemente sus estrategias y sus soluciones.

Si lo consideráramos pertinente en función de lo trabajado en el desarrollo, aprovechamos para recuperar la manera en que Scratch utiliza las coordenadas X e Y para ubicar los objetos y cómo podemos aprovecharlas para indicar el movimiento de esos objetos. Podemos también comparar esta idea con la estrategia de apuntar en dirección (■) y mover (■) pasos.

Recuperamos también la noción de evento como una herramienta de programación que nos permite crear programas interactivos.

Actividad 2
Agregamos obstáculos

Se aumenta el programa incluyendo obstáculos y retomando las nociones de alternativa condicional y sensores para ponerlas en práctica en Scratch.

Objetivo >

Se espera que las y los estudiantes:

- Recuperen la alternativa condicional y los sensores para detectar colisiones entre los objetos.
- Refuercen la idea de desarrollo incremental y de proyecto de programación.

Inicio >

El **propósito de este momento** es recuperar lo hecho en la actividad anterior y presentar la nueva consigna.

Orientaciones

Luego de recuperar brevemente el trabajo realizado, presentamos la consigna de esta actividad: **hacer que los cubos de hielo impidan el avance del personaje**. Es una buena oportunidad para recordar que la computadora no realizará acciones que no estén previstas en el programa: si no programamos qué debe suceder cuando el personaje toca un cubo de hielo, este continuará con su avance y el cubo de hielo no se comportará como un obstáculo. Entonces es parte de la tarea de programación prever este comportamiento en el programa.

Desarrollo >

El **propósito de este momento** es que las y los estudiantes resuelvan, con la mayor autonomía posible, un problema asociado al uso de la alternativa condicional por fuera de un desafío cerrado de programación.

Orientaciones

Invitamos a los grupos a agregar la nueva funcionalidad a su proyecto. Si trabajaron con la secuencia "¿Podemos programar otros personajes?", de

esta colección[2], podemos invitarlos a recuperar algunos de los desafíos de las tarjetas que podrían ser útiles para resolver un problema cuando dos objetos se tocan.

Después de que hayan tenido la oportunidad de explorar los bloques y ensayar soluciones, podemos orientar una reflexión común sobre los bloques ¿Tocando (Cubo de hielo)? o ¿Tocando color (■)? (categoría Sensores) y la alternativa condicional Si… entonces (categoría Control) con preguntas como las siguientes.

¿Qué sucede si el personaje toca un cubo de hielo? ¿Qué debería suceder? Si viene desde la izquierda, ¿para qué lado debería retroceder? ¿Y si viene desde la derecha? ¿Qué bloques dan información sobre si dos objetos se están tocando? ¿Qué bloques nos sirven para expresar que una acción debe realizarse cuando se cumple determinada condición?

Además de identificar que pueden recurrir a los bloques de sensores (como ¿tocando …?) en combinación con la alternativa condicional, deberán decidir en qué lugar del programa colocarlo. Para esto, podemos ayudarlos a identificar que la pregunta que responde este sensor tiene sentido inmediatamente después de que el personaje haya avanzado.

Para indicar el retroceso del personaje, podemos apelar a reproducir la estrategia utilizada para el avance.

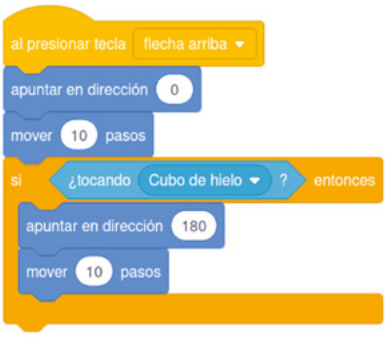
Si utilizaron los bloques apuntar… y mover…, existe una solución particularmente compacta y legible: utilizar números negativos en el bloque mover (■) pasos hace que el objeto se mueva en la dirección opuesta. Si las y los estudiantes están familiarizados con esta noción y la noción de signo, podemos alentar la reflexión para que identifiquen que retroceder es equivalente a avanzar, pero en sentido contrario y, por lo tanto, retroceder es equivalente a avanzar una cantidad negativa. Esta solución, además, permite definir un procedimiento para el retroceso que puede reutilizarse.

Durante el proceso de programación, podemos alentar a ordenar la estructura del programa mediante la definición de procedimientos (o "Mis bloques", en Scratch) para separar tareas y reutilizar fragmentos del programa si fuera posible. En este sentido, la solución para el movimiento con apuntar en dirección (■) y mover (■) pasos es particularmente adecuada.

Los siguientes fragmentos de soluciones posibles utilizan el sensor asociado al encuentro de objetos luego del movimiento del personaje para incorporar el retroceso.

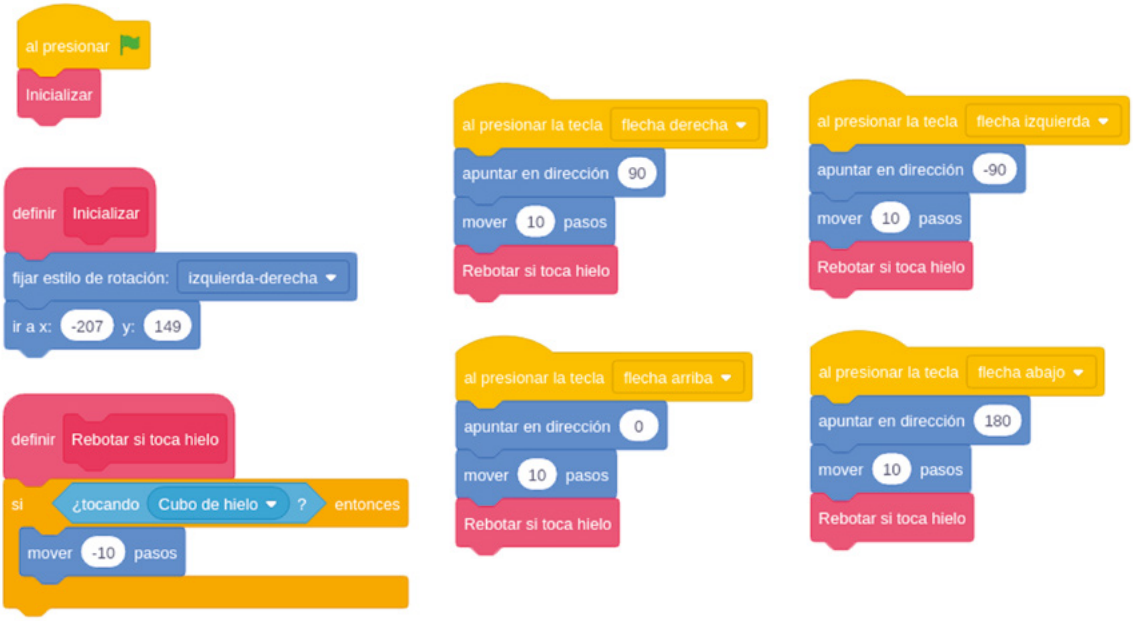
[2] Todas las secuencias de la colección se encuentran disponibles en el sitio curriculum.program.ar

El movimiento se realiza modificando las coordenadas del personaje.



El movimiento se realiza con los bloques apuntar en dirección (■) y mover (■) pasos.

La solución aprovecha el bloque apuntar en dirección (■) para expresar el retroceso con un valor negativo en el bloque mover (■) pasos.



Organización de una solución posible usando procedimientos.

Resuelta la consigna de la respuesta ante los obstáculos, continuamos con el ejercicio de desarrollo incremental, para lo que proponemos una nueva consigna: **hacer que el personaje retroceda al tocar el borde del escenario**. Invitamos a los grupos a explorar sensores disponibles para que reconozcan que pueden reproducir la estrategia utilizada ante otro elemento. Resolver un problema similar con una variante permite que refuercen las nociones vistas y adquieran autonomía.

Fragmentos de tres soluciones para que el personaje retroceda al querer avanzar más allá del borde del escenario, análogos a la estrategia elegida para programar el retroceso frente a los obstáculos.

Cierre >

El **propósito de este momento** es compartir las estrategias utilizadas y explicitar la experiencia de desarrollo incremental.

Orientaciones

Promovemos una puesta en común para que las y los estudiantes compartan sus soluciones y estrategias. Invitamos a conversar sobre las ventajas y las desventajas de cada estrategia. También se puede reparar en qué criterio fue el usado en los sensores (por ejemplo, si detectan contacto con un objeto o un color; en esta última, la estrategia puede tener el problema de que los cubos de hielo tienen tres caras visibles con tres tonos de color distintos, y si decidiéramos cambiar el tipo de obstáculo por otro de un color diferente afectaría negativamente el funcionamiento de todo el programa).

Detalle del cubo del hielo en el que se percibe que las tres caras tienen variaciones en su tonalidad.

Puesta en común

Por último, retomamos el recorrido realizado en lo que va del proyecto, identificando cada etapa y su contribución al producto final. En la primera, agregamos la posibilidad de interactuar para mover al personaje. En esta actividad agregamos el comportamiento de los obstáculos como tal, es decir, como objetos que limiten los movimientos de los personajes y, luego, incluimos también la limitación por los bordes del escenario.

Pensando en el producto final que exploramos al principio de la secuencia, promovemos que los grupos proyecten y prevean las funcionalidades que aún faltan para completar el proyecto.

Actividad 3
Agregamos enemigos

En esta actividad, se completa el proyecto. Las y los estudiantes aumentarán sus programas con la inclusión de un elemento tipo enemigo y un elemento como objetivo. Para concluir, se abstraen algunos problemas comunes al desarrollo de videojuegos y se refuerza la noción de proyecto y desarrollo incremental.

Objetivo >

Se espera que las y los estudiantes:

- Generalicen la noción de colisión entre objetos y las estrategias para detectarlas.
- Conciban un proyecto de programación como un proceso incremental formado por etapas que se van resolviendo sucesivamente.
- Identifiquen características comunes de los videojuegos y las asocien con estrategias de programación.

Inicio >

El **propósito de este momento** es presentar los desafíos que restan para completar el juego.

Orientaciones

Se presentan las últimas consignas para completar el juego:

- Hacer que los tiburones aparezcan durante un lapso de tiempo y luego desaparezcan por el mismo lapso, durante todo el desarrollo del juego.
- Hacer que Nacles retroceda hasta el comienzo del recorrido para reemprender la marcha si toca algún tiburón.
- Lograr que Nacles diga "¡Conseguí la manzana!", al llegar al objetivo.

Es importante reforzar la noción de desarrollo incremental y que el objetivo es el funcionamiento del videojuego completo. Para ello, podemos adelantar que, una vez que estén completas estas funcionalidades, podrán destinar el tiempo restante a aumentar o personalizar el juego, por ejemplo, agregando sonidos o haciendo que termine el programa o cambie el fondo cuando el personaje alcance el objeto.

¿Cuántas tareas tenemos que programar en cada consigna? ¿Alguna se puede subdividir? ¿Podemos hacer procedimientos? ¿Cuáles tareas consideran sencillas y cuáles complejas? ¿Existe un orden específico en el que debamos encarar las tareas?

Habilitamos un espacio de discusión para definir las tareas a resolver, priorizarlas y ordenarlas. Es importante señalar que, en principio, no hay un orden establecido en el que deban resolver las consignas y, por lo tanto, cada grupo podrá planificar su recorrido. Además, cada consigna es probable que se divida en tareas más simples que podrán reflejar en el programa con procedimientos.

Podemos invitarlos a que incluyan tareas de personalización en el último orden de prioridades para que puedan comenzar a ejercitar la gestión autónoma de los tiempos.

Desarrollo >

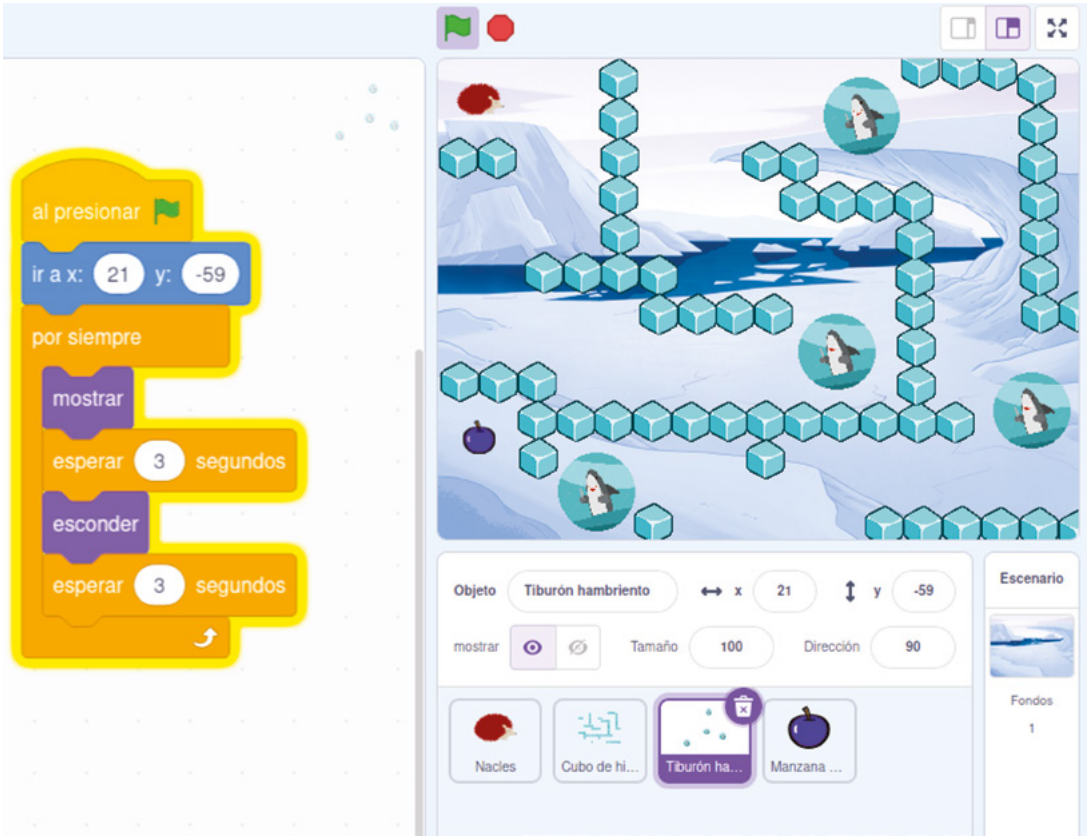
El **propósito de este momento** es brindar un espacio de trabajo autónomo para que los grupos completen sus proyectos de acuerdo con las tareas definidas.

Orientaciones

Para alentar el trabajo autónomo, prestamos atención a los avances y las dificultades de cada grupo para resolver dificultades o brindar pistas cuando sea necesario. A continuación brindamos algunas orientaciones sobre cada consigna.

Para lograr que los tiburones desaparezcan y aparezcan regularmente durante el desarrollo del juego, se puede motivar a los grupos a:

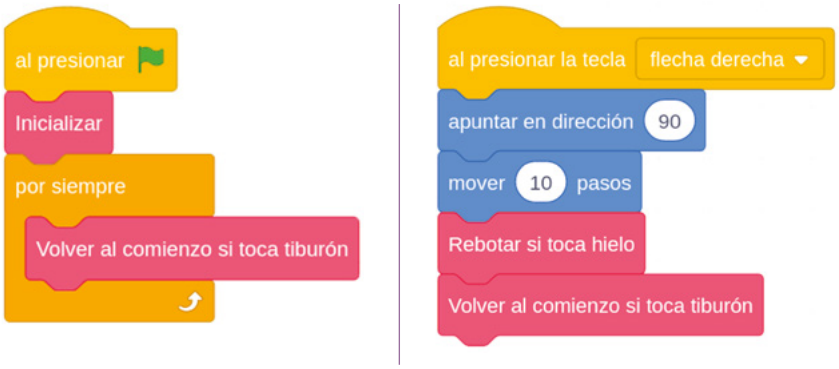
- Experimentar con el bloque Por siempre.
- Usar de los bloques mostrar y esconder.
- Recurrir al bloque esperar, cuya funcionalidad consiste justamente en detener la ejecución de las instrucciones por un lapso determinado de tiempo.
- Identificar que este comportamiento deben programarlo en los bloques del objeto Tiburón.



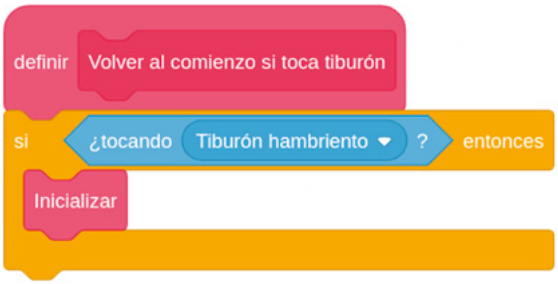
Animación de los tiburones según la consigna.

Para que Nacles retroceda al tocar un tiburón:

- Alentamos a los grupos a que identifiquen que este comportamiento es similar al que hicieron con Nacles y los bloques de hielo.
- Para que Nacles vuelva al inicio, podemos alentar a la reutilización del procedimiento Inicializar que está programado como parte del proyecto inicial, pero también a explorar el bloque ir a x (■) y (■) y el manejo de coordenadas (si se introdujo y explicó en la **Actividad 2**).
- La alternativa condicional que evalúa la colisión entre Nacles y los tiburones puede ubicarse dentro de los eventos de movimiento (al igual que hicimos para detectar la colisión con los bloques de hielo) o como parte de una repetición por siempre al iniciar el programa; este segundo caso podemos motivarlo a partir de pensar que es una pregunta que queremos que se realice constantemente, pues no sabemos cuándo puede llegar a ocurrir que se toquen.



Dos soluciones que utilizan el procedimiento Volver al comienzo si toca tiburón.



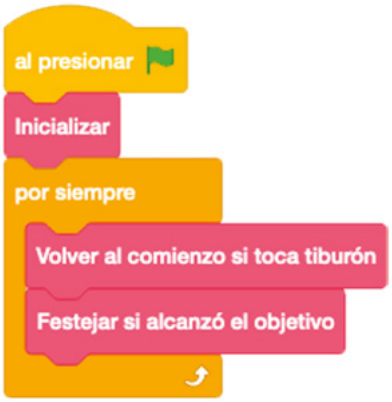
Procedimiento Volver al comienzo si toca tiburón para que Nacles vuelva al inicio al chocar contra un tiburón.

Para que Nacles festeje cuando alcanza la manzana:

- La clave es que los grupos identifiquen que este problema es un caso más de colisión, es decir, de detectar que dos objetos están en contacto y que, en consecuencia, se ejecute una acción. Podemos orientarlos recordándoles que ya resolvieron casos de colisión en la interacción con el hielo o con los bordes (y para los tiburones, si ya resolvieron esa consigna).
- Para determinar la colisión se recurre a una alternativa condicional con el sensor ¿tocando (Manzana polar)? Al igual que en el caso de los tiburones, podrán colocar la alternativa en los eventos de movimiento (en este caso, alentamos a que definan un procedimiento para evitar duplicar los bloques) o pueden hacerlo con repetición por siempre.

Procedimiento que evalúa si se cumplió el objetivo del juego para festejar.

Solución en la que se coloca el procedimiento Festejar si alcanzó el objetivo para que se evalúe cada vez que se produce un movimiento. Como en el caso anterior, hay que poner el bloque en los cuatro movimientos.

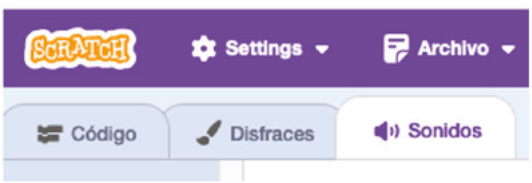


Otra solución, en la que el procedimiento se coloca en una repetición por siempre del programa principal.

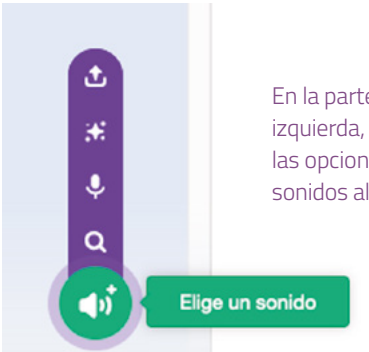
Para quienes completen estas funcionalidades básicas a tiempo, podemos mostrar alternativas para aumentar y personalizar el proyecto.

Para personalizar el juego, se pueden agregar sonidos. Por ejemplo, se puede reproducir un sonido a cada paso que dé Nacles o que se escuche una melodía cuando alcanza la manzana. Para esto:

- Se usan los bloques de la categoría Sonido.
- Para agregar más sonidos a nuestro proyecto, se puede ir a la pestaña "Sonido" (que se encuentra junto a la de "Código" y "Disfraces"). Desde allí se pueden agregar sonidos de una biblioteca, subir nuevos o grabar uno con el micrófono si el dispositivo que se está usando cuenta con uno.



La pestaña Sonidos.

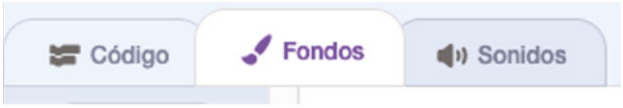


En la parte inferior izquierda, se encuentran las opciones para agregar sonidos al proyecto.

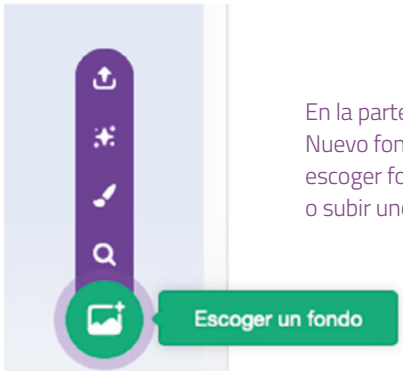
Para detener la ejecución del programa cuando se alcanza el objetivo, se utiliza el bloque detener todos en la categoría Control. Este detiene la ejecución de todos los bloques.

Para poner una pantalla de finalización al alcanzar la manzana, se puede cambiar la pantalla del juego modificando el fondo del escenario.

Seleccionamos el Escenario para modificar el fondo.



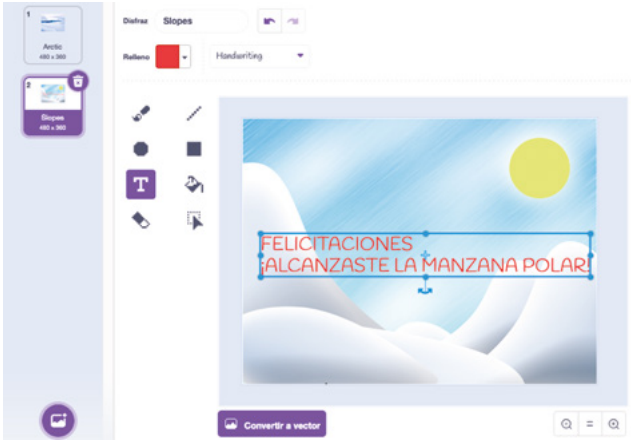
En la pestaña Fondos podemos agregar y editar imágenes para usar de fondo.



En la parte inferior, el botón Nuevo fondo nos permite escoger fondos de la biblioteca o subir uno de la computadora.

Podemos editar el fondo para modificarlo y agregarle texto. Para ello:

- Para cambiar el fondo, existe el bloque cambiar fondo a (■) en la categoría Apariencia.
- La próxima vez que se inicie el juego será necesario que se deshagan los cambios de apariencia y volver a mostrar los objetos y el fondo con el laberinto como estaban al comienzo. Esto depende del bloque Inicializar. Así se dimensiona su importancia para el correcto funcionamiento del juego.



El editor de fondos permite modificar un fondo prediseñado.

Cierre >

El **propósito de este momento** es identificar las características comunes de las soluciones programadas por los grupos y del proceso de desarrollo que puedan ser aplicables a futuros proyectos.

Orientaciones

Brindamos un espacio para que los grupos puedan presentar sus videojuegos. Cada grupo describe las particularidades de su videojuego y de la construcción de la solución.

En una puesta en común, los acompañamos para abstraer del proceso de programación los casos en los que definieron procedimientos, tanto para separar subtareas como para reutilizar fragmentos de programa.

También, es importante que logren **abstraer soluciones comunes a problemas frecuentes en los videojuegos**, por ejemplo, detectar y reaccionar cuando dos objetos se tocan (colisión) y , si lo hicieron, el uso del bloque Por siempre para verificar condiciones que pueden cumplirse en cualquier momento.

Puesta en común

Apuntando hacia una conclusión de la secuencia, se asocia la idea de un proyecto construido de manera incremental con la resolución de un problema dividiéndolo en subproblemas. Al igual que en los entornos cerrados (como Pilas Bloques), esto permite definir tareas, pensar estrategias y programar procedimientos que agreguen claridad al programa y que puedan ser reutilizados. En el trabajo en Scratch, estos subproblemas suelen convertirse en sucesivas etapas en las que se van "agregando" características hasta completar el objetivo. A este modo de trabajo le decimos "proyecto".

Para reforzar las ideas principales a partir de la generalización, podemos pensar en un videojuego hipotético (puede ser un auto de carreras que cuando toca el pasto se vuelve a meter a la pista, un barco que esquiva islotes, un superhéroe que transita un terreno peligroso, etc.) para identificar qué problemas deberíamos resolver, a qué problemas que ya resolvimos se parecen y cómo podríamos adaptar las soluciones que conocemos para ellos.

Secuencia didáctica 14

Guardamos información

Variables

¿Podemos ingresar información a nuestros programas? ¿Cómo hacemos para que guarden información durante su ejecución?

En esta secuencia propone una introducción a las variables como herramientas de programación para almacenar datos durante la ejecución de un programa desde una estrategia por indagación.

Actividad 1

Las y los estudiantes experimentan con los bloques *preguntar* y *respuesta* como una aproximación al ingreso y almacenamiento de información durante la ejecución de un programa.

Actividad 2

Continúan trabajando sobre el mismo proyecto, ahora con un nuevo desafío que requiere la introducción de variables.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: procedimientos, repeticiones, alternativa condicional, sensores, eventos, variables.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad.

Objetivos de aprendizaje

- Reconocer la necesidad de almacenar y consultar información para resolver un problema.
- Utilizar expresiones y comandos de variables para almacenar y consultar información durante la ejecución de un programa.

Saberes previos de CC

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, procedimientos, repeticiones, alternativa condicional, sensores.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad, reutilización del código.

Materiales necesarios

- Dispositivos con Scratch 3 instalado o acceso a su versión en línea https://scratch.mit.edu.ar/
- Proyecto de Scratch con el videojuego para explorar (El castillo exclusivo.sb3).

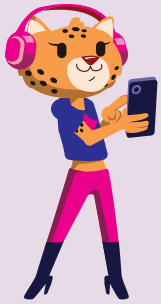
Actividad 1
Bienvenida al castillo

En esta actividad las y los estudiantes se aproximan a los bloques preguntar y respuesta a partir de analizar y aumentar un programa en el que un personaje requiere una contraseña para entrar a un castillo.

Objetivo >

Se espera que las y los estudiantes:

- Reconozcan la posibilidad de que los programas almacenen información durante su ejecución.
- Utilicen bloques para almacenar y recuperar información en la construcción de un programa.

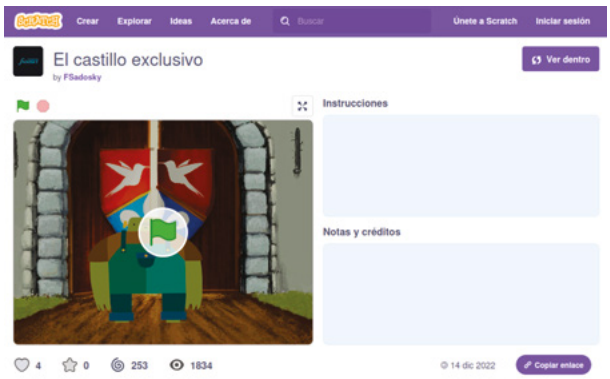


Inicio >

El **propósito de este momento** es presentar los bloques preguntar y respuesta como un mecanismo para ingresar y almacenar información durante la ejecución de un programa.

Orientaciones

Invitamos a las y los estudiantes a formar grupos heterogéneos[1] de trabajo. Cada grupo ingresará al proyecto de Scratch *El castillo exclusivo* en línea ingresando a ***https://scratch.mit.edu/projects/777062677*** o abriendo el archivo del proyecto El castillo exclusivo.sb3, disponible en los materiales de esta secuencia.

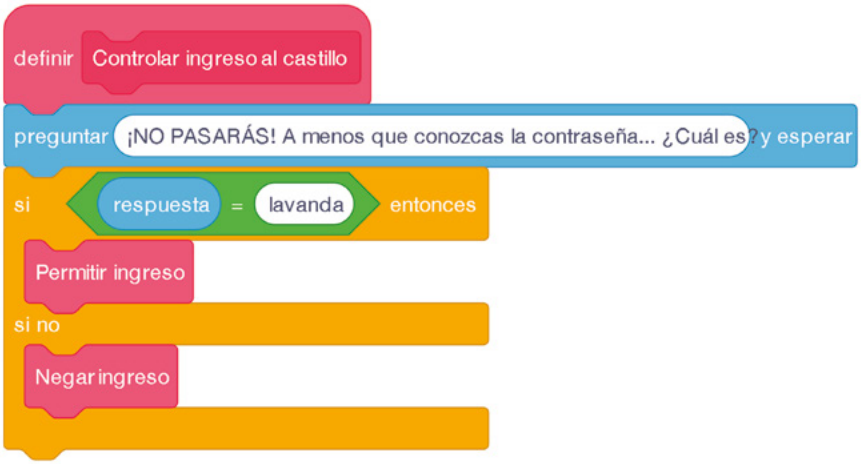


Proyecto *El castillo exclusivo* en la galería de Scratch.

[1] Se pueden consultar dinámicas lúdicas para el armado de grupos heterogéneos en "Enseñar computación desde la mirada de la Educación Sexual Integral (ESI)", 10.

El objetivo del juego es ingresar al castillo y, para ello, hay que informar la contraseña correcta al guardia que custodia la entrada. Les proponemos a los grupos que jueguen unas rondas para conocer el juego y piensen cómo podrían averiguar la contraseña. Posiblemente, en un principio, prueben contraseñas al azar y obtengan resultados negativos.

El siguiente paso es investigar el programa "por dentro" para encontrar cuál es la contraseña válida. Para eso, recordamos que existe el botón "Ver dentro". Se espera que los grupos reconozcan que la respuesta se encuentra en el procedimiento Controlar ingreso al castillo, en el que se utiliza una alternativa condicional para permitir o negar el ingreso al castillo a partir de una condición que compara el contenido de la respuesta introducida por el usuario o la usuaria con la palabra "lavanda". Esta es la palabra que debemos escribir para que el guardia nos deje ingresar al castillo. Cuando todos los grupos lo hayan hallado, promovemos una puesta en común.



El procedimiento Controlar ingreso al castillo controla el ingreso a partir de una condición sobre la respuesta.

¿Qué hicieron para descubrir la contraseña? ¿Qué parte del programa tuvieron que revisar en detalle? ¿Algo les facilitó descubrir qué parte del programa había que analizar? ¿Hay bloques que no hayan visto en otros programas de Scratch? ¿Cuál es su propósito?

De las experiencias relatadas por las y los estudiantes resaltamos que:

- Podemos entender mejor cómo funciona un programa analizando cómo está construido, aún cuando no sea de nuestra autoría.
- Los procedimientos y su nomenclatura facilitan la interpretación de un programa para otras personas. (Esta noción se presentó previamente en esta colección y, en esta secuencia, se ve con un caso concreto).

- Los nuevos bloques preguntar y respuesta plantean un flujo de información particular: el bloque preguntar permite al usuario o la usuaria ingresar un texto, y usarlo en el programa mediante el bloque respuesta (por ejemplo, para compararlo con el texto de la contraseña definida).
- Estos bloques cumplen una función nueva para las y los estudiantes: el **almacenamiento de información** (el texto ingresado en un momento se almacena para estar disponible en el programa más adelante).

Desarrollo >

El **propósito de este momento** es plantear un nuevo desafío que se resuelva mediante el uso de los bloques preguntar y respuesta, e introducir el operador unir para juntar cadenas de caracteres.

Orientaciones

Proponemos a los grupos que modifiquen el programa para que el guardia pregunte el nombre y lo use para saludar, y luego pida la contraseña.

La secuencia esperada inicia con el personaje preguntando a las y los jugadores el nombre. Luego, se habilita un campo para ingresar el nombre mediante el teclado. A continuación, el personaje emite un saludo personalizado con el nombre que se ingresó.

Damos a los grupos un tiempo para resolver por sus medios esta consigna, prestando especial atención a las dificultades que puedan encontrar en el uso de los nuevos bloques. También, probablemente sea necesario presentar el operador unir para elaborar un único mensaje a partir de dos textos.

Una solución para lograr que el personaje solicite el nombre a las y los jugadores y luego lo utilice en el saludo.

Antes de invitar a los grupos a resolver el desafío, podemos sugerirles que lo asocien a un procedimiento nuevo, y aprovechamos la ocasión para repasar las virtudes del uso de los procedimientos que se repasaron en el inicio.

Solución al desafío, con el conjunto de bloques para saludar definidos como procedimiento.

Cierre >

El **propósito de este momento** es explicitar el proceso de ingreso y almacenamiento de información en la ejecución del programa y reconocer otras situaciones de uso de artefactos computacionales en los que sucede lo mismo.

Orientaciones

Invitamos a los grupos a compartir sus estrategias de solución y los orientamos para lograr que abstraigan la función de los bloques preguntar y respuesta en términos del ingreso y almacenamiento de información.

¿Qué pasa cuando se ejecuta el bloque preguntar? ¿Qué pasa después de que las o los jugadores ingresan el texto? ¿Cómo se accede a esa información en el programa?

Para reforzar esta idea, invitamos a que piensen otras situaciones de uso con artefactos computacionales que tengan un comportamiento análogo. Un ejemplo habitual es el ingreso con usuario y contraseña, pero también el funcionamiento de la tarjeta de transporte o cualquier otra de identificación donde la información almacenada se coteja contra un listado de datos para su validación.

Actividad 2
Te llamaré por tu nombre

Un nuevo desafío en la actividad del castillo requiere usar en más de una oportunidad el nombre solicitado al comienzo y, por lo tanto, motiva el uso de variables.

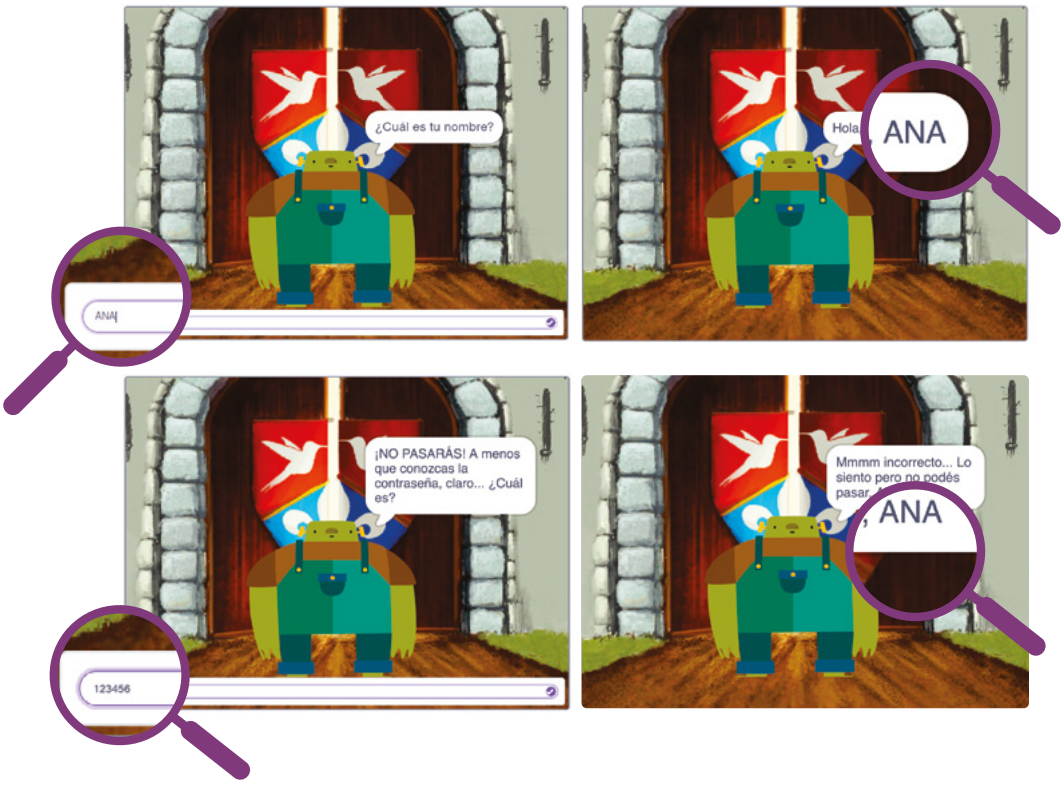
Objetivo >

Se espera que las y los estudiantes conozcan la noción de variables como una herramienta para almacenar y recuperar información durante la ejecución de un programa.

Inicio >

El **propósito de este momento** es presentar un problema que requiere almacenar información para ser resuelto.

Comentamos a las y los estudiantes la nueva consigna. Además de saludarnos con nuestro nombre, tenemos que lograr que el personaje use nuestro nombre en la respuesta que nos brinda sobre si nos autoriza o no a entrar al castillo. Y es importante que lo haga preguntándonos el nombre una sola vez. Para ilustrar este comportamiento, podemos mostrar cómo funciona la solución completa.



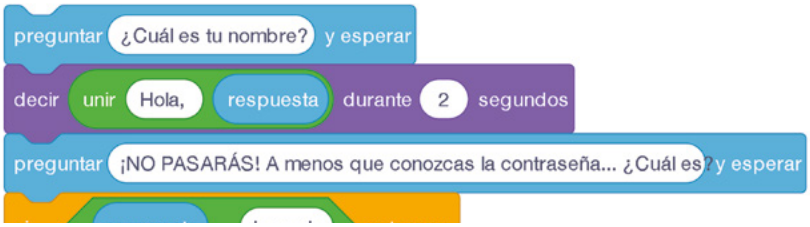
Invitamos a que ensayen una solución, resaltando que el personaje no puede volver a preguntar el nombre. **Alentamos la experimentación para que descubran por su cuenta que, con esta restricción, el problema no se puede resolver utilizando únicamente los bloques** preguntar y respuesta.

Para resolver este problema las y los estudiantes cuentan con herramientas para ingresar un dato y almacenarlo (preguntar y esperar), y para recuperarlo durante la ejecución del programa (respuesta). Sin embargo, al querer utilizar el nombre luego de preguntar por la contraseña podrán notar que la información del nombre ya no está disponible en el bloque respuesta. A partir de esta situación podemos promover un momento de análisis.

¿Qué fue lo que sucedió? ¿Son suficientes las herramientas que tenemos a disposición? ¿Qué efecto tiene el bloque preguntar sobre el bloque respuesta? ¿Cuánta información puede guardar, a la vez, el bloque respuesta?

Este planteo, nos pone frente a un problema que no tiene solución con las herramientas disponibles hasta el momento. Esto habilita la formulación de hipótesis sobre lo que está pasando e intentar comprender mejor cómo funcionan las herramientas que estamos utilizando.

Presentamos un problema, intentamos resolverlo con las herramientas que conocemos hasta el momento, pero no fue posible. Este obstáculo motiva la reflexión para identificar que para resolver el problema es necesario **almacenar** el nombre que se ingresa para utilizarlo cuando se pida la contraseña. Así es que las **variables** entrarán en escena y las presentaremos como una herramienta para resolver un **problema concreto** (que, comprendemos mejor, pues ya intentamos resolverlo con lo que conocíamos): **almacenar información durante la ejecución de un programa**.



Si se usara el bloque preguntar.... y esperar en dos ocasiones, al almacenar la segunda respuesta se borrará la anterior. Por lo tanto, no es válida esta estrategia.

A partir de la instancia de experimentación, hacemos una puesta en común para que las y los estudiantes expliquen el problema que encontraron y expliciten la consecuente necesidad de almacenar el nombre de alguna otra manera que no sea mediante el bloque respuesta.

Desarrollo >

El **propósito de este momento** es presentar la noción de variable y su uso para almacenar y recuperar información durante la ejecución de un programa.

A partir de la dificultad expuesta en el inicio, presentamos las variables como una herramienta de los lenguajes de programación para almacenar información durante la ejecución de un programa.

Mostramos el proceso de creación y sus bloques asociados. Podemos proyectar el proceso para toda la clase.

Comenzamos con la creación de una variable. Para eso, en la categoría Variables, encontramos el botón "Crear variable".

Comienzo del proceso de creación de una variable

A continuación, debemos elegir un nombre para la variable. Como en el caso de los procedimientos, es importante utilizar una denominación representativa que identifique qué información va a contener la variable para cuidar la legibilidad del programa.

Como va a almacenar el nombre de una persona, se eligió "nombre de la persona" para denominar a esa variable.

Una vez creada la variable, aparecen bloques para operar con ella.

¿Cómo funcionan las variables? ¿Qué bloques asociados encontramos en el entorno? ¿Qué propósito tienen? ¿Cómo las utilizamos en nuestros programas?

Estas preguntas habilitan la exploración del entorno, de los bloques (comandos y expresiones) asociados a las variables: para asignarle un nuevo valor (dar o fijar), o para alterar su valor actual (cambiar o sumar). De esta manera, a través de un recorrido marcado por una **indagación** guiada, introducimos una nueva herramienta y construimos junto a las y los estudiantes una definición de su propósito, funcionamiento y uso en un programa.

En particular, nos importa explicitar el funcionamiento de los bloques fijar [nombre de la persona] a ( ) (almacena un determinado valor en la variable) y nombre de la persona (al igual que el bloque respuesta, nos permite acceder a la información almacenada en la variable).

Bloques para operar con variables.

A continuación, invitamos a los grupos a que utilicen estos nuevos bloques para resolver el desafío. Recorremos los puestos de trabajo y brindamos asistencia para que superen rápidamente las cuestiones operativas referidas a la creación de las variables y el uso de sus bloques asociados.

El bloque que utilizamos para acceder a un dato almacenado que ingresa la usuaria o el usuario (como, en este caso, nombre de la persona) tiene una forma redondeada, sin parte para encastre, que evidencia que no puede utilizarse por sí solo, sino que es una parte a incluir en otros bloques; esto se debe a que es un bloque que representa información y no una acción. Esto también está reflejado en el nombre de la variable: en general, utilizaremos sustantivos para expresar aquello que está almace-

nado, a diferencia de los nombres de los procedimientos que suelen ser verbos que expresan la acción que representa el procedimiento.

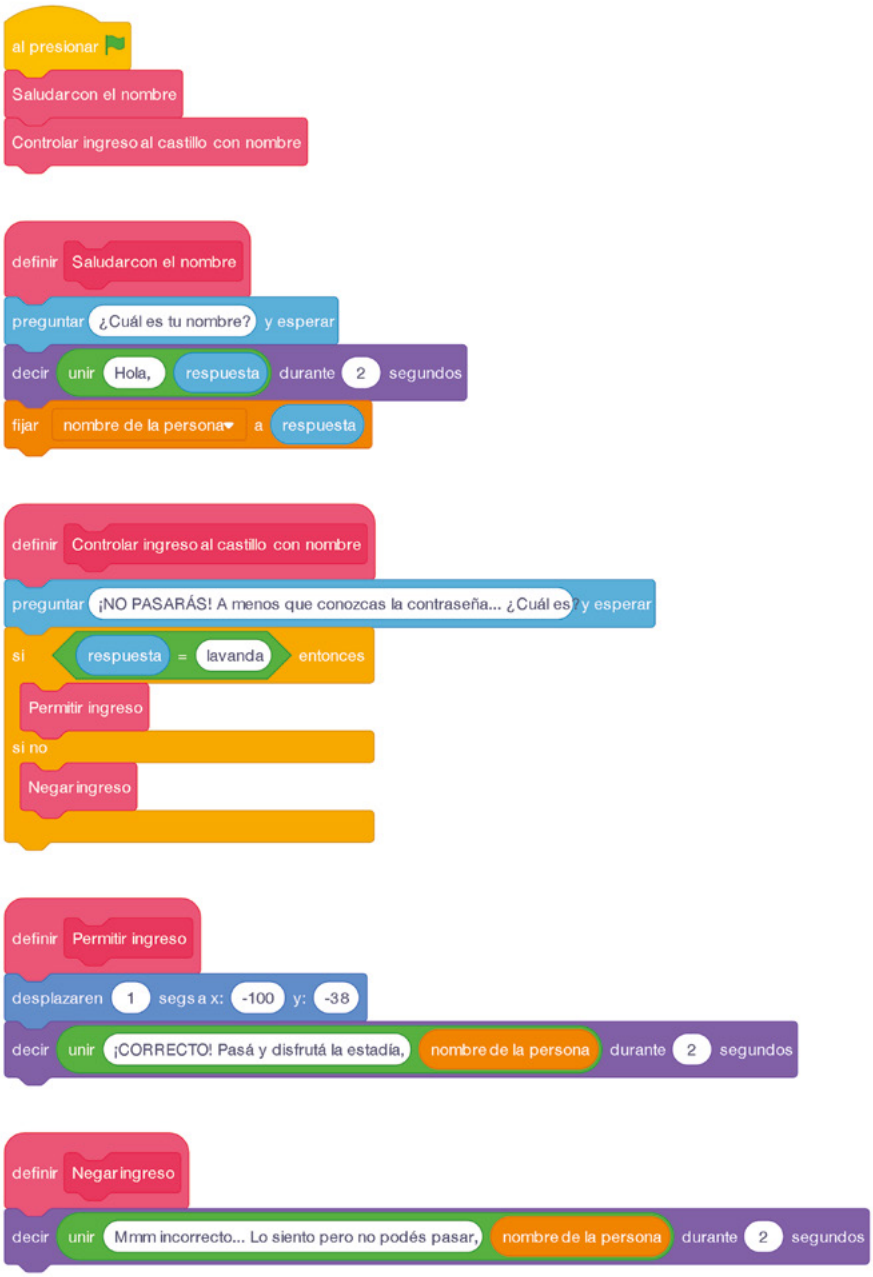
Nos interesa que los grupos se concentren en cómo integrar esos bloques al programa para resolver el problema. En este sentido, es importante que comprendan dónde deben colocar el bloque fijar (si lo utilizan después de preguntar la contraseña, seguirán teniendo el mismo problema). Podemos guiar esta exploración con preguntas para explicitar por qué deben colocarlo en ese lugar y no antes o después.

El casillero que aparece a la izquierda de los bloques de variables y de respuesta es una herramienta útil para analizar la información almacenada en las variables durante la ejecución del programa. Al tildarlo, aparece en la esquina superior izquierda del escenario cada variable con su contenido.

Tildar el casillero junto al bloque de variables nos permite visualizar su contenido en el escenario y su modificación a lo largo de la ejecución. En este caso, aún no se ha ejecutado el programa y, por lo tanto, las variables están vacías o contienen valores por defecto.

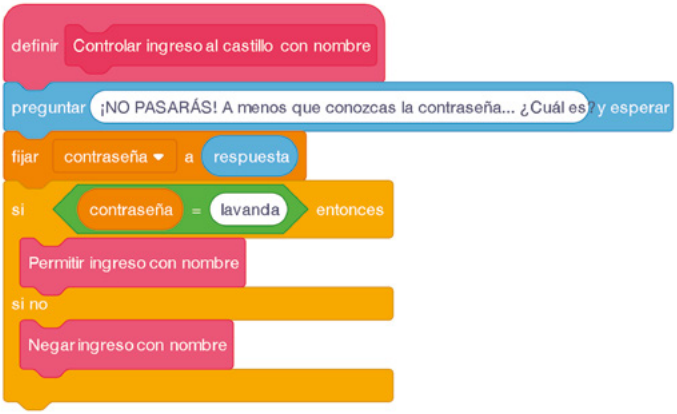
La solución consiste en almacenar la respuesta del usuario o la usuaria en una variable antes de volver a usar el bloque preguntar... y esperar.

Usamos el bloque fijar para almacenar el nombre ingresado por el usuario o la usuaria en una variable y, luego, lo utilizamos en las respuestas en los procedimientos correspondientes, de una manera análoga a como se hizo con el bloque respuesta en la **Actividad 1**. De esta manera, podremos acceder al contenido almacenado en la variable utilizando el bloque nombre de la persona, independientemente del uso que hagamos del bloque preguntar.



Procedimiento Controlar el ingreso al castillo de una posible solución. El nombre de la persona se almacena en la variable nombre de la persona para ser utilizado en los procedimientos Permitir ingreso y Negar ingreso; se utiliza el bloque de respuesta de la segunda pregunta para validar la contraseña.

También se puede utilizar una variable para la contraseña. Utilizar variables en vez del bloque respuesta también es una manera de mejorar la legibilidad del programa.



Procedimiento Controlar ingreso que aprovecha al máximo el uso de variables para mejorar la legibilidad del programa.

En una puesta en común, compartimos las estrategias de cada grupo haciendo foco en el uso que hace cada una de las variables. También señalamos el uso del bloque unir que nos permite concatenar varios textos; esta función es muy útil para construir mensajes que involucran varios datos distintos (por ejemplo, el contenido de una variable y textos fijos como "Hola").

Cierre >

El **propósito de este momento** es recuperar la experiencia de la secuencia para reforzar la idea de que hay situaciones o problemas en las que los programas necesitan recibir y almacenar información durante su ejecución; y arribar a conceptualizar a las variables como una herramienta de los lenguajes de programación para realizar esas operaciones.

Habilitamos un espacio de discusión y puesta en común. Podemos motivar el intercambio planteando un problema similar al resuelto en la actividad. Por ejemplo, podemos mostrar un fragmento de programa y hacer preguntas como las siguientes:

Analicen esta solución que propuso un/a compeñero/a para que un personaje salude a dos personas por su nombre. ¿Cumple el objetivo? ¿Por qué? Si no lo cumple, ¿qué indicaciones le darían a su compañero para que cumpla el objetivo?

Un ejemplo posible para reflexionar sobre el uso de variables.

A partir de las respuestas de las y los estudiantes, establecemos analogías con el trabajo hecho durante la última actividad para explicitar cuál fue el problema inicial, por qué no pudieron resolverlo en el primer intento y cómo lograron solucionarlo con la nueva herramienta.

Finalmente, repasamos los conceptos que esperamos que las y los estudiantes hayan comprendido.

- **Al crear una variable, hay que nombrarla**. Elegir denominaciones claras para variables y procedimientos aporta claridad y expresividad al programa, y, por lo tanto, mejora su legibilidad. A la creación de una variable se la denomina "declaración de la variable".

Almacenar un dato en una variable es una **acción** que se llama "asignación" (decimos "asignar un dato a una variable") y, por lo tanto, se realiza con un **comando**; por eso, el bloque fijar o dar en Scratch tiene la misma forma que otros bloque que representan acciones (como decir o mover) y se realizan una después de la otra.

*Piensen en los bloques que conocieron en esta secuencia (**preguntar, respuesta, fijar y nombre de la variable**). ¿Qué parecidos encuentran? ¿Pueden explicar el funcionamiento de unos a partir de otros?*

Con estas preguntamos buscamos que las y los estudiantes reconozcan que el bloque preguntar es una acción que incluye la asignación del texto ingresado por un/a usuario/a a una variable y que el bloque respuesta nos permite acceder al contenido de la respuesta, al igual que el bloque nombre de la persona nos permitió acceder al nombre almacenado en la variable.

Cuando hacíamos dos preguntas. ¿Qué pasaba con el valor de la primera respuesta cuando ingresábamos la segunda? ¿Cuál era el contenido de la variable?

El hecho de que se perdiera la primera respuesta nos da la pauta de que las variables solo pueden almacenar un dato por vez: durante la asigna-

ción, se descarta el valor que contiene la variable (si es que existe) y se almacena el valor nuevo. Si necesitamos almacenar más datos, tendremos que definir más variables.

Una vez más, hemos completado **el recorrido completo de una actividad por indagación** para presentar una noción puntual de programación, esta vez en un entorno abierto. En el inicio de la actividad **planteamos una situación problemática** concreta y habilitamos un espacio de **exploración**, que nos permitió concluir (a partir de comprobar en la propia experiencia que no era posible resolverlo con las herramientas conocidas) la **necesidad** de almacenar información para **resolver el problema**. A partir de esta motivación, y recién en este momento del desarrollo, **presentamos las variables** y mostramos cómo usarlas. A continuación, habilitamos una nueva instancia para **experimentar** con la nueva herramienta y usarla para resolver el problema. Finalmente, y no menos importante, recuperamos el trabajo de la actividad para **aclarar definiciones y reforzar conceptualizaciones**. Reflexionamos a partir de la experiencia, los problemas encontrados y las soluciones ensayadas para construir la noción de variable y explicar cómo se usan. Por lo tanto, el momento de **cierre** es donde suceden las **discusiones teóricas** más intensas.

Secuencia didáctica 15

Programamos nuestro videojuego

Variables, interactividad y un videojuego abierto

Hacemos nuestro primer videojuego desde cero, diseñando personajes, fondos, objetos y más. ¿Cómo haremos para contar los puntos? ¿Y para que el personaje atrape un objetivo?

En esta secuencia, se aborda el proyecto de programar un videojuego en Scratch en el que los personajes atrapan objetos para ganar puntos. El proyecto se construye de forma incremental a partir de la resolución de una secuencia de pequeños desafíos. En el proceso, se recupera el concepto de variable, que se utilizará para almacenar los puntos de cada jugador, y se integran herramientas de programación presentadas en las secuencias anteriores de esta colección.

Actividad 1

Siguiendo una serie de desafíos, las y los estudiantes construyen el escenario de su videojuego y completan una primera versión en la que el objetivo se reubica aleatoriamente al ser alcanzado por el personaje.

Actividad 2

Las y los estudiantes recuperan el trabajo con variables para sumar puntos cada vez que el personaje captura el objetivo.

Actividad 3

Las y los estudiantes trabajan autónomamente para agregar un segundo personaje, replicando el trabajo de las actividades anteriores. También se proponen desafíos opcionales para construir condiciones de finalización del juego.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: procedimientos, repeticiones, alternativa condicional, sensores, eventos, variables.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad.

Objetivos de aprendizaje

- Reconocer la necesidad de conservar, acceder y modificar información.
- Utilizar expresiones y comandos de variables para conservar, acceder y modificar información durante la ejecución de un programa.
- Comprender la noción de contador como una estrategia abstracta para el problema de llevar un recuento durante la ejecución de un programa.
- Generalizar la noción de evento y asociarla con la combinación de la repetición por siempre y una alternativa condicional.
- Conocer un proceso de desarrollo incremental y sus ventajas.

Saberes previos de CC

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: comandos primitivos, procedimientos, repeticiones, alternativa condicional, sensores, eventos, variables.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad, reutilización del código.

Materiales necesarios

- Dispositivos con Scratch 3 instalado o acceso a su versión en línea https://scratch.mit.edu.ar/
- Proyecto de Scratch con el videojuego completo de ejemplo (Valentino.sb3).

Actividad 1
Nuestro videojuego

Se presenta el proyecto, haciendo énfasis en el modo de construcción incremental en el que en cada paso se agregan nuevas funcionalidades y se construye una primera dinámica con un personaje y un objetivo que se reubica automáticamente al ser alcanzado.

Objetivo >

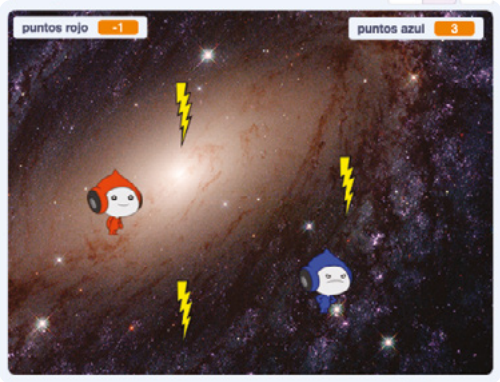
Se espera que las y los estudiantes:

- recuperen la noción de desarrollo incremental.
- construyan una estrategia de solución para implementar un evento genérico (más allá de los provistos por el entorno).

Inicio >

El **propósito de este momento** es presentar el modo de trabajo y construir una primera versión del videojuego en la que el personaje se desplace por el escenario.

Organizamos la clase en grupos heterogéneos pequeños[1]. Les pedimos que ingresen a Scratch. Para motivar el trabajo y visualizar el resultado final, una opción es mostrar una versión de un videojuego terminado similar al que programarán. Podemos proyectar la versión terminada e invitar a los grupos a que lo jueguen. Esta versión se encuentra en el archivo Valentinos.sb3 que acompaña esta secuencia.



Vista de un proyecto terminado, en el que dos personajes deben capturar objetos para sumar puntos.

[1] Se pueden consultar dinámicas lúdicas para el armado de grupo heterogéneos en "Enseñar computación desde la mirada de la Educación Sexual Integral (ESI)", 10.

Luego de la presentación del ejemplo, introducimos la primera etapa del proceso incremental: crear un proyecto en Scratch en el que un personaje sume puntos al capturar objetos. Todos los demás aspectos, funcionales y visuales (el escenario, los personajes y los objetos), serán elegidos libremente por los grupos. Dado que el desarrollo del proyecto probablemente abarque varias clases, reforzamos la importancia de ir guardando los proyectos (ya sea en la cuenta en línea o los archivos .sb3 en la computadora).

Desafío 1. Elegir **el personaje y el escenario**.

Para esto, pueden utilizar los personajes y los escenarios que ofrece Scratch pero también utilizar ilustraciones propias. En cuanto al funcionamiento, el/la usuario/a deberá poder mover al personaje en las cuatro direcciones con el teclado. Podemos referir al trabajo realizado en la **Actividad 1** de la secuencia didáctica "¿Podemos programar otros personajes?", de esta colección.

Enfatizamos que trabajaremos de manera incremental, es decir, agregaremos funcionalidades en sucesivas etapas, de manera que se obtenga **una versión funcional (aunque incompleta) del juego al término de cada etapa**. Por ejemplo, comenzaremos por definir el personaje, esto supone colocar el personaje y dejar completa la funcionalidad de moverlo por el escenario al pulsar teclas del teclado.

Comenzamos a construir el proyecto. Indicamos a los grupos que ingresen a Scratch y abran un proyecto nuevo.

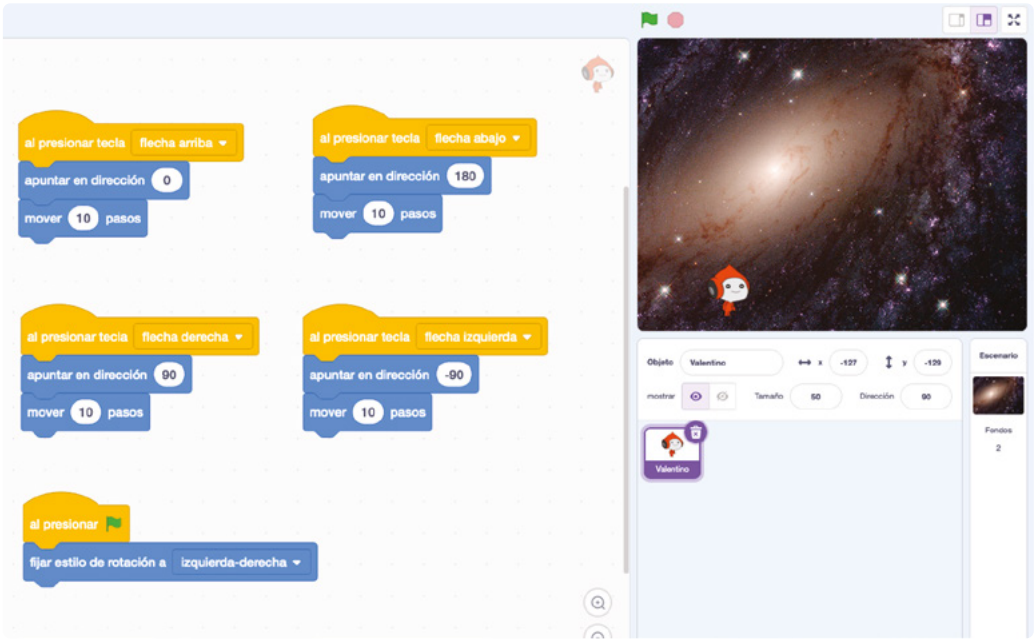
Material de referencia para docentes

Articulación con educación artística y prácticas del lenguaje

Scratch tiene un editor de imágenes incorporado en el que se pueden dibujar personajes y escenarios, pero también ofrece la posibilidad de incorporar archivos de imágenes (lo que permite utilizar editores de imágenes externos más complejos e incluso incorporar ilustraciones analógicas digitalizadas). Es importante tener en cuenta que al principio pueden utilizarse ilustraciones sencillas e ir completándolas en paralelo mientras se avanza con la programación para incorporarlas al proyecto cuando se consideren listas.

En esta misma línea, podemos proponer también pensar la narrativa del juego. Un videojuego cuenta una historia, que puede ser una aventura. Por lo tanto, podemos promover que construyan la historia, y le pongan nombre al personaje, que tenga una personalidad determinada y un propósito en el mundo-escenario.

Se espera que al resolver el desafío obtengan un escenario similar al siguiente.



Etapa inicial: el personaje controlado por el usuario/a y el escenario.

Dependiendo de la estrategia elegida para desplazar al personaje, puede ser necesario usar el bloque Fijar estilo de rotación izquierda-derecha para que el personaje no quede "patas para arriba" al moverlo.

Desarrollo >

El **propósito de este momento** es proponer una secuencia de consignas o desafíos que alienten a construir, gradualmente, una estrategia de solución para detectar colisiones utilizando la repetición "por siempre".

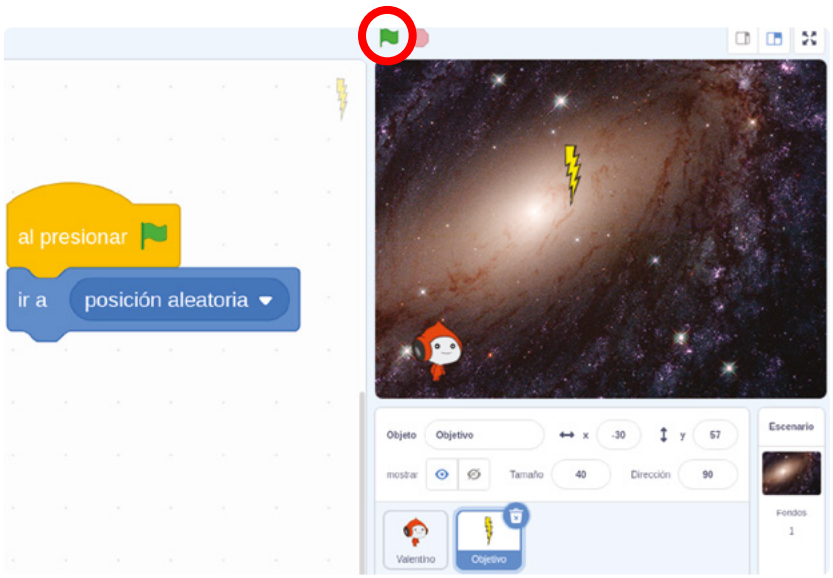
Continuamos con la programación del videojuego. Avanzamos a una segunda etapa en el proceso incremental. Anticipamos a las y los estudiantes que resolveremos consignas breves que consisten en pequeños agregados o modificaciones. En esta línea, nos va a interesar cómo resuelven cada consigna y, por lo tanto, alentamos a que registren el proceso de resolución de cada una, en particular, qué pruebas hicieron, a qué conclusiones llegaron, qué decisiones tomaron en función de esas conclusiones y cómo construyeron la solución.

Desafío 2. Incorporar el **objetivo**, es decir, un objeto que será el que deba recoger el personaje. Deberá aparecer en una posición aleatoria al presionar la bandera verde del entorno de Scratch.

Es probable que necesiten reducir el tamaño del objetivo elegido y el personaje para que haya más distancia entre ellos y mejore la jugabilidad.

Invitamos a los grupos a programar esta nueva funcionalidad y alentamos a probar y documentar cada cambio que hagan en el proyecto. Dejamos que trabajen autónomamente y prestamos atención a eventuales dificultades.

Es posible que requieran utilizar el bloque ir a posición aleatoria. Si tienen dificultades, podemos recordarles que se abordó en la tarjeta "Mover aleatoriamente e ir a posición" de la secuencia didáctica "¿Podemos programar otros personajes?", de esta colección.



El videojuego con un objetivo (el rayo) que se ubica en una posición aleatoria al comenzar.

Desafío 3. Programar el objetivo para que al presionar la bandera verde **cambie 20 veces de lugar aleatoriamente, cada un segundo**.

Damos un tiempo para que pongan manos a la obra en la resolución de esta consigna.

El propósito de esta consigna es recuperar los bloques de espera, pero, sobre todo, plantear un desafío que se resuelva a partir de una modificación de un programa ya existente con bloques que ya conocen. En este sentido, es importante que las y los estudiantes identifiquen que deben agregar al programa que ya tienen un bloque de repetición y un bloque de espera para resolver el desafío.

Una solución posible para el desafío 3.

Desafío 4. Programar el **objetivo para que aparezca en lugares aleatorios** del escenario durante toda la ejecución del juego, cambiando de lugar una vez por segundo.

Para afrontar esta resolución deberán buscar un nuevo bloque que asegure que el proceso se repetirá de manera constante mientras el juego se encuentre activo. Al igual que en el desafío anterior, nos interesa plantear la solución como una modificación al programa existente. Para esto, pueden reemplazar el bloque de repetición por el bloque por siempre. Si no lo conocieran o necesitaran una pista, invitamos a explorar la categoría Control.

Una posible solución para reubicar el objetivo durante toda la ejecución del juego.

Otro intento de solución es conservar la repetición simple, pero con una cantidad "alta" de repeticiones. Esto parece resolver el problema mientras la cantidad de repeticiones sea suficientemente alta para que atrapemos el objetivo antes de que termine de moverse, pero no resuelve el desafío por completo. Podemos hacer esta consideración en este momento o avanzar con el siguiente desafío, que no puede resolverse sin el uso de la repetición por siempre.

Resuelta esta consigna, es un buen momento para reconocer los sucesivos pasos dados; podemos apoyarnos en notas que hayan tomado las y los estudiantes en la documentación del trayecto.

¿Qué problemas intermedios resolvieron para lograr el videojuego que programaron? ¿Qué pedía cada desafío? ¿Cómo los consiguieron atender? ¿Qué tuvieron que agregar o modificar en el programa? ¿Reconocen por qué elegimos esta secuencia de desafíos?

El objetivo de esta puesta en común es identificar la noción de desarrollo incremental, que consiste en agregar una funcionalidad a cada paso, trabajando sobre un resultado previo. En nuestro caso, además, era posible resolver cada paso modificando algún aspecto del programa, lo que nos obligó a analizar nuestras propias soluciones para identificar qué parte debíamos modificar. Esto fue posible dado que cada desafío planteaba un objetivo con una diferencia pequeña respecto del anterior, lo que lo hacía más fácil de atender. Esto fue parte de una planificación previa, una manera de organizar el trabajo de programación.

El próximo paso nos acercará al funcionamiento esperado del juego.

Desafío 5. Hacer que el objetivo cambie su posición aleatoriamente **cada vez que sea atrapado por el personaje**, en lugar de cambiar cada segundo.

Esta tarea tiene una dificultad adicional a las anteriores, ya que requiere abordar las colisiones de una manera diferente a cómo se hizo en las secuencias "¿Podemos programar otros personajes?", "Introducción a Scratch" y "Programamos el personaje de un videojuego". Interactividad y eventos" de esta colección.

Invitamos a las y los estudiantes a ensayar autónomamente una solución. Posiblemente descubran que no pueden programar la reubicación del objetivo desde los eventos de movimiento del personaje, ya que los bloques que mueven al personaje y el que reubica el objetivo refieren a objetos distintos.

Ejemplo de una solución incorrecta: detectar la colisión en los eventos de movimiento del personaje no permite desplazar el objetivo porque tanto el bloque mover como ir a posición aleatoria actúan sobre el personaje.

Una vez que los grupos hayan reconocido la dificultad, podemos concluir entre todas y todos que en Scratch no pueden combinarse comandos que actúen sobre objetos distintos porque cada uno cuenta con un espacio de programación diferente.

Esta dificultad plantea uno de los problemas centrales que queremos abordar en esta secuencia: diseñar una estrategia de solución para implementar eventos genéricos. Por lo tanto, en este momento, es importante habilitar la exploración y el ensayo de soluciones, brindando orientaciones que encaucen la reflexión, pero sin adelantar la solución final ni conceptualizaciones excesivas.

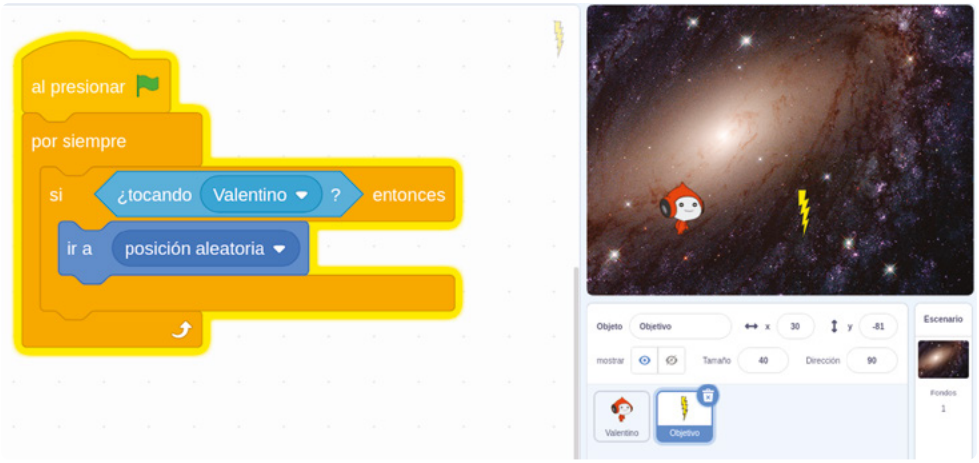
¿Cómo hicieron en otros proyectos para detectar que un objeto estaba tocando a otro? ¿Dónde ubicaron esos bloques? ¿Por qué no pueden ubicar esos bloques en un evento del teclado? ¿Qué acción debe suceder cuando se toquen los objetos? ¿Qué bloque van a utilizar para eso? ¿Sobre qué objeto debe realizarse esa acción? ¿En el espacio de qué objeto debemos colocar ese bloque?

Una orientación posible es recuperar la necesidad de utilizar una alternativa condicional con el sensor que detecta si se está tocando otro objeto, como hicieron en secuencias anteriores. Lo siguiente es atender a la ubicación de los bloques. Para esto, podemos ayudar a las y los estudiantes a identificar que, dado que la acción que deberá suceder cuando los objetos se toquen recae sobre el objetivo (este debe reubicarse en una posición aleatoria, como lo hace el bloque ir a posición aleatoria), deberán agregarlos como parte de los bloques de este objeto.

El otro aspecto de la solución a construir es el temporal.

¿Sabemos en qué momento hay que evaluar la condición? ¿Cuántas veces debe suceder?

Estas reflexiones apuntan a explicitar que la pregunta por si se están tocando el personaje y el objetivo debe hacerse todo el tiempo y, por lo tanto, una solución posible es colocar la alternativa condicional dentro del bloque por siempre que se abordó en el desafío anterior.



Una solución posible, en el programa del objetivo, que utiliza el sensor para detectar si está tocando el personaje.

Cuando esté resuelto este desafío, realizamos una puesta en común para explicitar esta **nueva estrategia para detectar colisiones** que combina la alternativa condicional, un sensor y la repetición por siempre.

¿Qué bloques utilizaron para programar la colisión entre objetos? ¿En qué se diferencia de como lo hicieron en otras oportunidades?

Cierre >

El **propósito de este momento** es recuperar la experiencia de la actividad para identificar el proceso de desarrollo incremental y abstraer la estrategia para detectar las colisiones como un caso nuevo de evento.

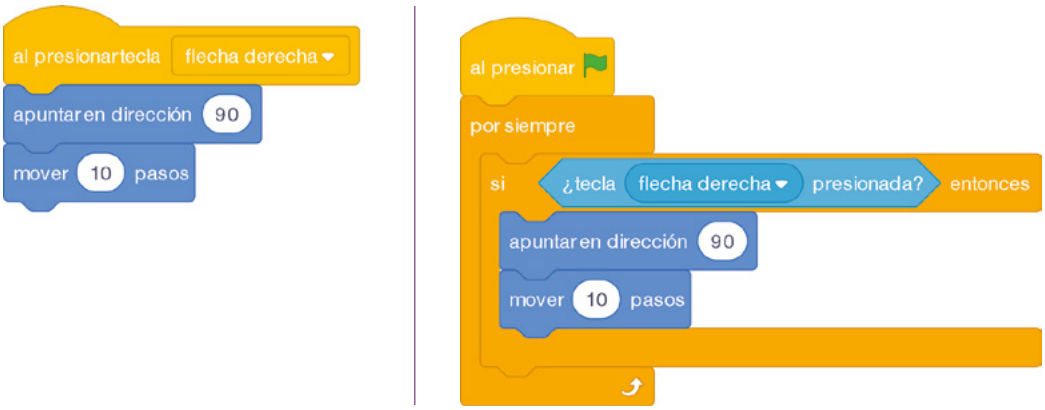
Orientaciones

Invitamos a las y los estudiantes a recuperar todos los pasos hechos, ayudándonos con las notas que han tomado. Relevamos los pasos dados en la actividad, las funcionalidades incorporadas en cada momento y qué cosas nuevas aprendieron al resolver las consignas que se presentaron.

Realizamos una nueva puesta en común para abstraer la estrategia de combinar la repetición por siempre con la alternativa condicional para detectar una situación que puede ocurrir en cualquier momento del juego.

¿Qué estrategia usaron para reubicar el objetivo cuando era capturado? ¿Qué otros bloques usamos en los videojuegos para indicar que cierta acción debe realizarse cuando ocurre algo? ¿Cómo hicieron para que el personaje se mueva a la izquierda cuando el usuario o la usuaria presione la flecha izquierda? ¿Encuentran parecidos con la estrategia que utiliza la repetición por siempre? ¿Podrían programar ese comportamiento utilizando la misma estrategia que usaron para reubicar el objetivo?

Orientamos la puesta en común para que comprendan que la estrategia que combina la alternativa condicional con la repetición por siempre es equivalente a un evento, ya que ambos permiten establecer qué debe suceder en el programa cuando ocurre algo en particular (por ejemplo, al presionar una tecla o cuando dos objetos se tocan). Los bloques de evento son una herramienta que nos brinda el lenguaje para expresar mejor nuestros programas, pero internamente funcionan con la estrategia de la repetición por siempre.



Dos fragmentos de programa que tienen el mismo efecto.

Actividad 2
Argegamos un contador de puntaje

Se analizan videojuegos para identificar el almacenamiento de información. Como conclusión, surge el uso de las variables como herramienta para completar el desafío de la actividad: agregarle puntaje al juego.

Objetivo >

Se espera que las y los estudiantes:

- Recuperen las variables como herramientas para almacenar información y usarlas para construir un contador.
- Reconozcan un contador como una estrategia general.

Inicio >

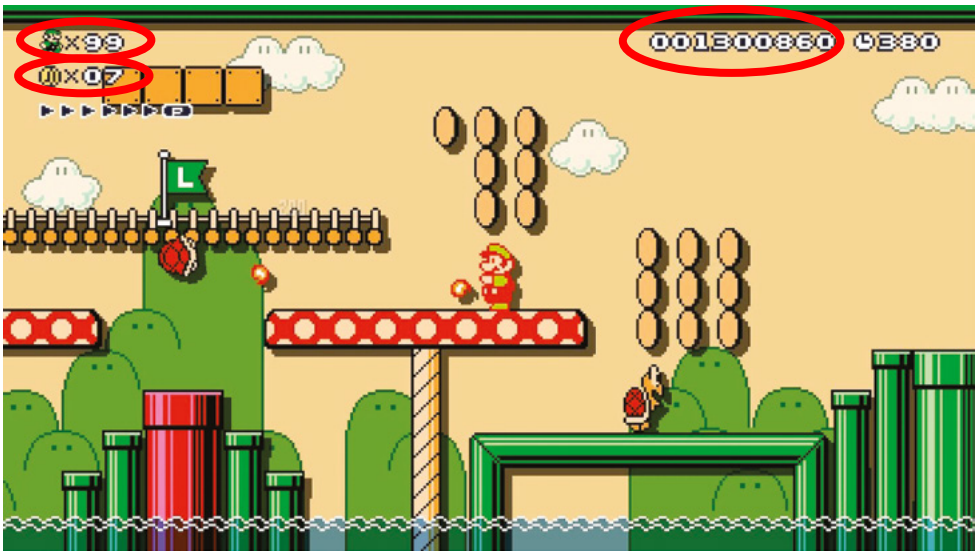
El **propósito de este momento** es presentar el almacenamiento de información durante la ejecución de un programa.

Orientaciones

Preguntamos a las y los estudiantes acerca de qué videojuegos conocen, a cuáles juegan habitualmente, si tienen alguno favorito, si conocen juegos *vintage*, etc. Si contamos con conexión a internet, podemos proponerles que busquen dos imágenes por grupo de la interfaz de alguno de los videojuegos que recuerdan. Si no hay conexión, podemos proyectar o compartir algunas que hayamos seleccionado previamente.

Proponemos analizar entre todas y todos las capturas para identificar qué información guardan los juegos durante su funcionamiento y qué valores almacenan y utiliza.

¿Qué cosas que ven en la pantalla les parece que requieren que se almacene información? ¿Qué datos almacena cada uno? ¿Cuándo varían?



Algunos elementos visibles que reflejan la necesidad de almacenar información en el videojuego *Super Mario Bros.*

Por ejemplo, en la captura de Super Mario Bros., se identifica que el puntaje, la cantidad de monedas y las vidas disponibles es información que existe durante todo el juego y, por lo tanto, el programa debe conservarla. En este caso son tres valores numéricos que cambian cuando el personaje se encuentra con distintos objetos. Existen otros tipos de información que también se conservan durante la ejecución, como el tiempo y las habilidades que adquiere temporalmente el personaje al recolectar algunos objetos. Estos aspectos son menos explícitos que los primeros, por los que alentamos su hallazgo, pero enfatizamos que nos concentraremos en los primeros, que son los que se asemejan a lo que deberán atender.

Desarrollo >

El **propósito de este momento** es brindar a las y los estudiantes una instancia de trabajo autónomo con acompañamiento docente para que experimenten con los bloques de variables para construir un contador.

Orientaciones

Planteamos la consigna y brindamos un espacio de exploración.

> ***Desafío 6***. Agregar puntaje al videojuego: **cada vez que el personaje atrapa el objetivo se suma un punto**.

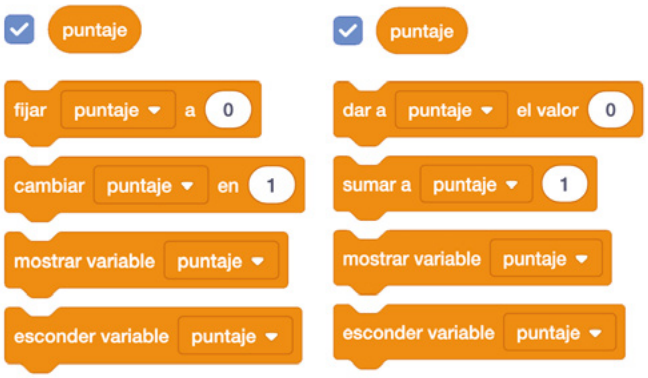
Si fuera necesario, orientamos con preguntas la recuperación del trabajo del inicio (los videojuegos almacenan información durante el juego) y la función de las variables como herramientas para almacenar información

(como se trabajó en la secuencia de esta colección "Guardamos información. Variables").

A continuación de esta exploración, concluimos que podremos usar una variable para almacenar el puntaje; recordando la importancia de nombrarlas de manera descriptiva, podemos llamarla puntaje, por ejemplo.

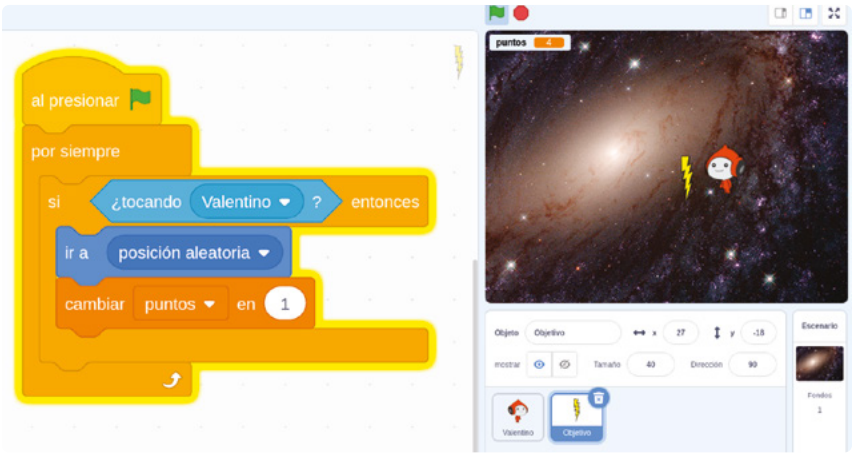
El próximo paso es analizar los bloques disponibles de la categoría **Variables**. Dependiendo de la traducción utilizada de Scratch, será más o menos directo cuál es el bloque que podemos utilizar para resolver el problema.

Dependiendo del nivel de avance de los grupos y su familiaridad con estos bloques, podemos realizar una exposición o habilitar un espacio para que avancen con la programación y nos consulten si encuentran dificultades.



Bloques de variables en la traducción al español latinoamericano (izquierda) y de España (derecha).

Habilitamos nuevamente la exploración con el objetivo de que las y los estudiantes identifiquen que pueden usar el bloque cambiar o sumar a. Si no identifican en qué parte del programa colocar el bloque, podemos orientarlos con preguntas para que reconozcan que ya cuentan con un fragmento de programa que detecta cuando el personaje atrapa el objetivo: la repetición por siempre que agregaron en el desafío anterior.



Una primera solución posible para acumular puntaje.

Si bien agregar el bloque cambiar resuelve parte del problema, tiene un inconveniente que podemos resaltar observando qué sucede con los puntos cuando volvemos a ejecutar el programa y qué esperaríamos que pase. Esto apunta a identificar la necesidad de establecer (o fijar) el valor del puntaje en cero al comenzar la ejecución del juego. Nuevamente, invitamos a analizar los bloques de variables en busca de una solución.

Con esta solución el puntaje vuelve a cero al reiniciar el juego.

Cierre >

El **propósito de este momento** es que las y los estudiantes recuperen la experiencia de lo realizado para reforzar la conceptualización de las variables como herramientas de programación para almacenar información y abstraer la estrategia para contar los puntos.

Orientaciones

Realizamos una puesta en común sobre la experiencia de resolver el desafío del desarrollo.

¿Cuál era el problema central del desafío de esta actividad? ¿Qué herramienta de programación usaron para resolverlo? ¿Por qué? ¿De qué manera? ¿Cuál es la estrategia para contar?

Nos interesa relacionar el problema de contar los puntos con la necesidad de almacenar información y, en particular, explicitar la estrategia utilizada para contar los puntos. Nos interesa identificar que usamos una variable, cuyo valor se establece en cero al comenzar el programa (a esto le decimos "inicializar la variable") y que se aumenta en uno cada vez que se captura el objeto; señalamos el uso de los bloques fijar o establecer y cambiar en o sumar a y comparamos su función. A esta construcción la llamamos **contador**.

Actividad 3
Añadimos jugadores y jugadoras

Resueltas las fases anteriores (un objetivo que cambia de posición y un contador que suma un punto por cada vez que el personaje colisiona con el objetivo), se modifica el videojuego para que pueda ser jugado por más de un/a usuario/a.

Objetivo >

Se espera que las y los estudiantes:

- Profundicen la conceptualización del uso de variables para construir contadores.
- Incorporen la detección de colisiones utilizando la repetición por siempre.

Inicio >

El **propósito de este momento** es presentar la nueva consigna y establecer relaciones con el trabajo realizado en la **Actividad 2**.

Orientaciones

Presentamos la consigna para aumentar el videojuego realizado en la actividad anterior:

> ***Desafío 7***. Agregar un **nuevo personaje**, controlado por un/a **segundo/a jugador/a** que competirá con el original por alcanzar el objetivo en simultáneo.

A partir de lo hecho en la actividad anterior, charlamos entre todas y todos para detallar las tareas que requiere el nuevo desafío:

» Incorporar un nuevo personaje al juego que se mueva con el teclado, utilizando teclas diferentes a las del personaje existente (por ejemplo, A-S-D-W).

» Hacer que el objetivo se reubique cada vez que sea alcanzado por cualquiera de los dos personajes.

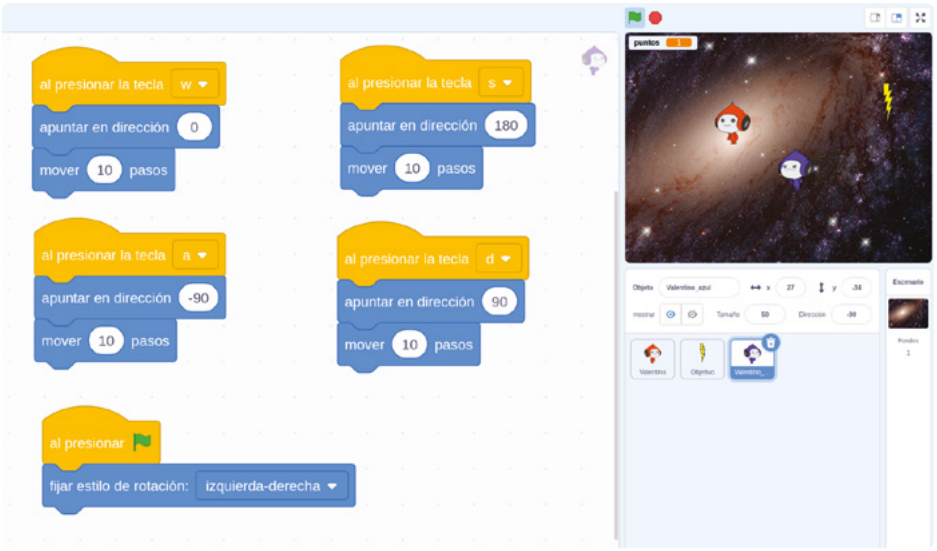
» Contar los puntos del nuevo personaje.

Desarrollo >

El **propósito de este momento** es brindar una instancia de trabajo autónomo para que las y los estudiantes recuperen las estrategias vistas en la actividad anterior.

Invitamos a los y las estudiantes a continuar con el desarrollo incremental de su videojuego. Alentamos a **identificar similitudes** con los problemas que ya resolvieron en la **Actividad 2** y a **recuperar las soluciones que construyeron** en esa oportunidad. También, alentamos a que identifiquen **qué partes del programa original deben modificar** para atender al nuevo desafío. Propiciamos la reflexión sobre la función de cada parte del programa recordando en qué paso y por qué fue agregada.

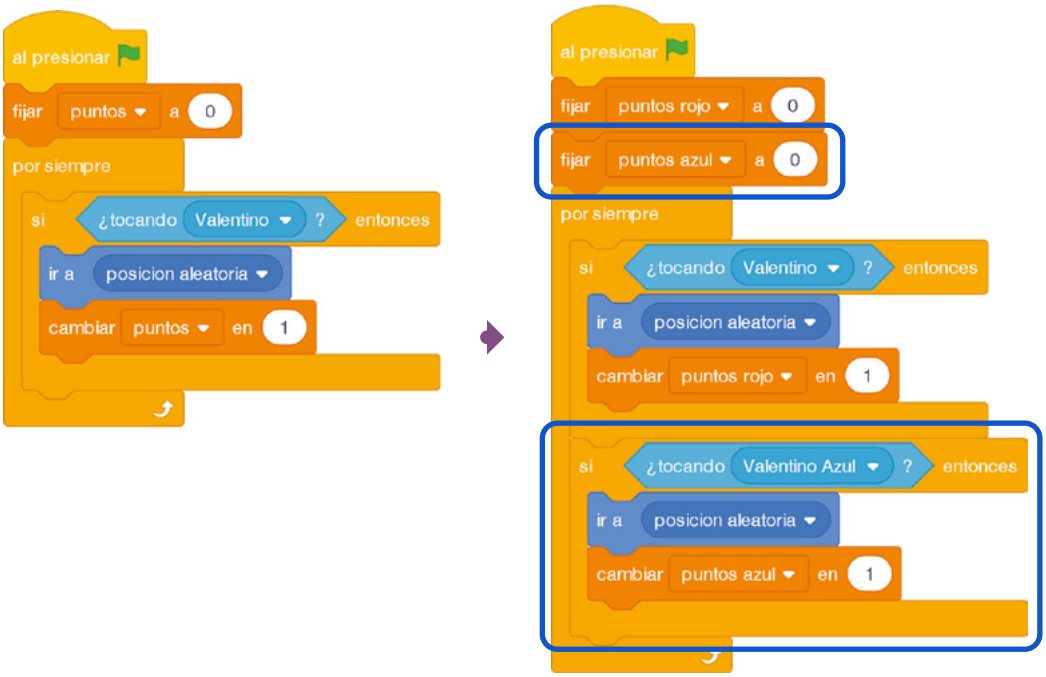
Para hacer que el nuevo personaje (Valentino Azul) se mueva, podemos replicar los eventos del personaje original (Valentino) y modificar las teclas a la que responden.



Agregado de un personaje "Valentino Azul" con su propia interactividad.

Para contar los puntos de otro/a jugador/a, necesitaremos una nueva variable para agregar un contador al programa. Esta estrategia también es réplica del contador que construimos en la **Actividad 2**.

La reubicación del objetivo también se resuelve de manera análoga. Estas similitudes son intencionales y nos interesa que las y los estudiantes puedan identificarlas para guiar la programación.



Agregados en el programa del Objetivo para cumplir con la nueva consigna. Para cambiar el nombre de la variable "puntos" por "puntos rojo" se usa la función "Renombrar" que aparece haciendo clic con el botón derecho del mouse sobre el bloque de la variable.

Cuando los grupos hayan alcanzado la consigna, realizamos una puesta en común.

¿Identificaron problemas parecidos a otros que ya habían resuelto durante este proyecto? ¿Se ayudaron con estos para resolver los problemas nuevos? ¿Hubiera sido más o menos complejo programar desde el comienzo los dos personajes? ¿Por qué?

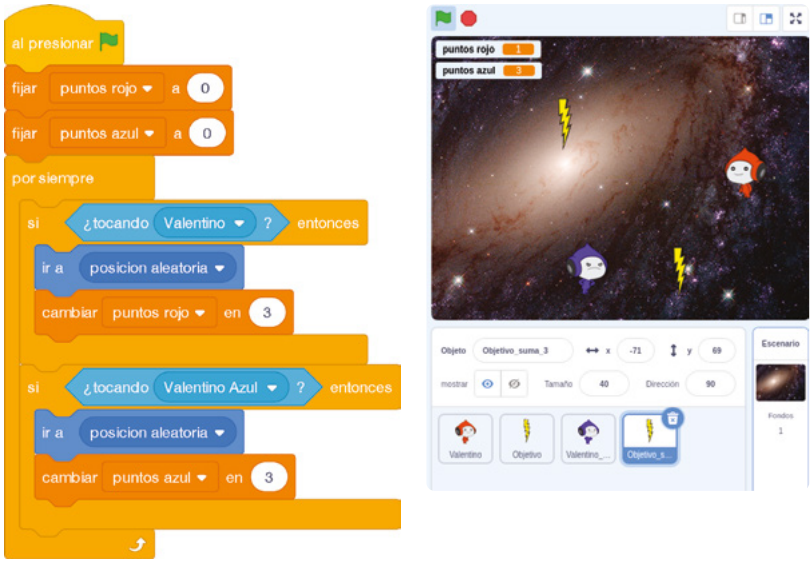
Identificamos entre todas y todos las ventajas de haber comenzado programando uno de los personajes: los problemas fueron más acotados y, una vez resueltos, pudimos agregar el segundo personaje replicando las soluciones que ya conocíamos.

Presentamos el siguiente desafío a los grupos. Le damos espacio para que trabajen autónomamente aprovechando las estrategias que ya conocen.

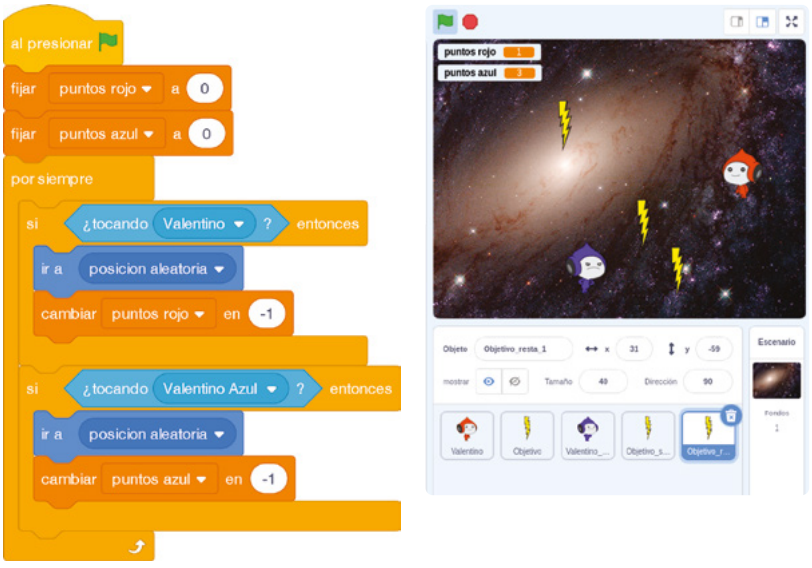
Desafío 8. Incorporar dos nuevos objetivos: uno que sume 3 puntos al ser capturado y otro que reste un punto. Los dos, al igual que el original, deben cambiar de posición aleatoriamente al ser capturados. Adicionalmente, hacer que los tres objetivos luzcan idénticos para que las y los jugadores no sepan con cuál colisionan.

Podemos brindar un acompañamiento más intenso en el agregado de uno de los objetivos y luego un espacio para que ensayen y generalicen con el otro.

Para resolver el desafío, se puede usar la función **Duplicar** haciendo clic derecho sobre el objeto, que genera un objeto idéntico con los mismos bloques que el original que, a partir de modificaciones, puede convertirse en la solución a la consigna.



Solución posible para el objetivo que suma 3 puntos.



Solución posible para el objetivo que resta un punto.

Resuelta esta nueva consigna, hacemos una puesta en común para que las y los estudiantes cuenten cómo lograron atender al desafío, haciendo especial énfasis en cómo aprovecharon la solución existente y qué modificaciones tuvieron que hacerle.

Dado que las variables son accesibles por todos los objetos del programa, cabe la pregunta sobre dónde colocar los bloques para restablecer las variables a 0 cuando se inicia el programa. En el ejemplo, replicamos la inicialización en los tres objetivos, pero esto no es necesario. Tampoco es recomendable en programas de mayor complejidad para evitar bloques con comportamientos superpuestos. De hecho, alcanza con colocar estos bloques en un objeto, asociados al evento de inicio (la bandera verde). Para reflejar que son acciones que no refieren a ningún objeto en particular, sino al contexto del videojuego, podemos ubicarlos como **bloques del Escenario**.

Para quienes deseen continuar, estas son algunas propuestas adicionales, que también se construyen con contadores y condiciones por siempre. El objetivo es hacer que el juego se detenga cuando sucede alguna de las siguientes condiciones.

Desafío opcional. Si alguno de los personajes alcanza (o supera) una cantidad definida de puntos (por ejemplo, 15), el juego termina y se informa que ese personaje es el ganador.

Se busca detectar cuándo el valor del puntaje tiene un valor determinado. Se debe colocar en algún bloque de repetición por siempre una alternativa condicional que evalúe esta condición y contenga la acción de informar quién es el ganador y termine el juego. Debemos replicar este comportamiento para cada jugador/a. Es importante alentar la definición de procedimientos para cuidar la legibilidad de programa.

Una solución posible con procedimientos para verificar los puntajes.

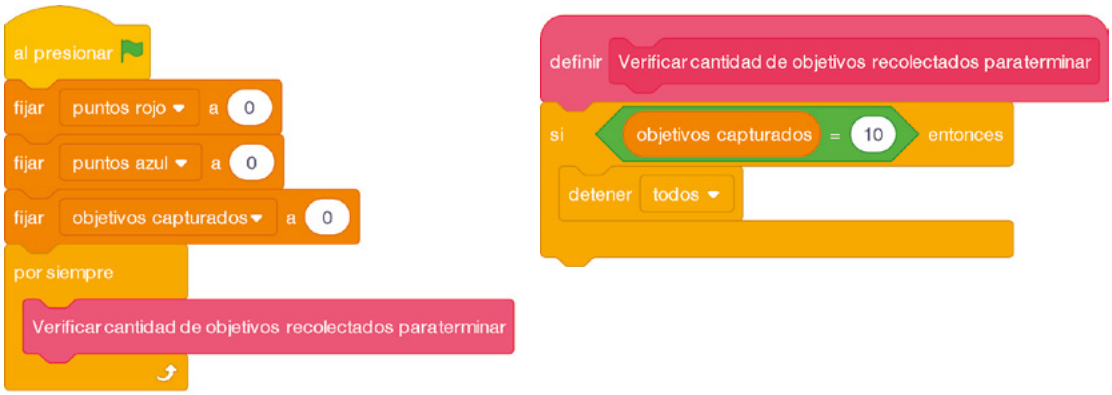
Desafío opcional. Cuando se captura una cantidad de objetivos entre ambos personajes (por ejemplo, 10), el juego termina.

Este desafío requiere llevar un recuento distinto: la cantidad de objetivos recolectados en total. Luego, el primer paso para resolver este problema es llegar a esta conclusión y, por lo tanto, construir un nuevo contador. El valor de este contador comienza en 0 al comenzar el programa y aumenta en uno cada vez que alguno de los personajes captura un objetivo. Por un lado, se deben separar la inicialización de las variables en un objeto. Por otro, colocar el bloque cambiar en el programa de cada uno de los objetivos, dentro de la alternativa condicional que detecta la colisión para cada uno de los jugadores.

En esta solución posible, se inicializan las tres variables en algún objeto (izquierda) y luego, en el programa de cada objetivo, se agrega, en la colisión con cada uno de los personajes, el bloque que incrementa la cuenta de objetivos capturados (derecha).

Resuelto el problema de contar, resta verificar si la cantidad de objetivos recolectados es la esperada.

Para esto, podemos recurrir nuevamente a una alternativa condicional dentro de una repetición por siempre. Para mejorar la legibilidad, definimos un procedimiento especialmente para esto. Es importante señalar también que, en este caso, no es relevante en qué objeto se coloca esta verificación, pues todos están ejecutando su repetición por siempre constantemente. Podemos colocarla junto con la inicialización de las variables para reflejar que no son acciones que involucren a ningún objeto en particular.



Agregamos la verificación de la cantidad de objetivos capturados dentro de una repetición por siempre.

Desafío opcional. Cuando alguno de los personajes captura una cantidad definida de objetivos, termina el juego.

Esta consigna es similar a la anterior, pero requiere llevar dos cuentas por separado: los objetivos recolectados por el jugador rojo y los recolectados por el jugador azul.

Cierre >

El **propósito de este momento** es brindar una instancia de reflexión y metacognición para reforzar la conceptualización de una estrategia para detectar colisiones y generalizar la noción de eventos, el uso de variables como contadores y el proceso de desarrollo incremental y sus ventajas.

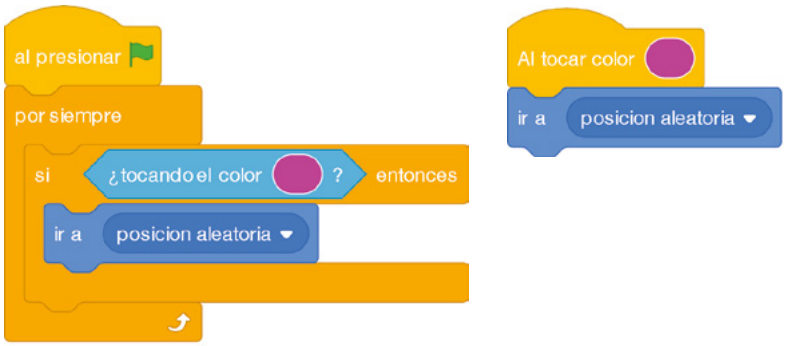
Orientaciones

Se ha abordado una secuencia muy prolífica e intensa. Aprovechamos este trabajo como insumo para reforzar conceptualizaciones y revisar los aprendizajes logrados.

¿Qué problemas tuvieron que resolver durante la secuencia? ¿Cuáles les parece que son problemas comunes, es decir, que probablemente encontrarían al desarrollar otros videojuegos? ¿Qué estrategias utilizaron para resolverlos? ¿Qué herramientas de programación usaron? ¿Qué aprendieron a partir de enfrentarse con cada uno de estos problemas?

En este intercambio, nos interesa abstraer dos estrategias de solución particulares para generalizarlas. Por un lado, el problema de reubicar los personajes al capturar el objetivo motivó el uso de la **repetición por siempre combinada con la alternativa condicional** y un sensor para realizar una acción cuando sucede algo en particular en un momento cualquiera de

la ejecución del juego. Como un primer paso, podemos preguntarnos qué situaciones podríamos detectar con esta estrategia, cómo lo haríamos y cómo lo pensaríamos como un evento; por ejemplo, podemos hacer que, cuando el personaje toque un objeto de color magenta, se reubique en una posición al azar.



A la izquierda, un fragmento de programa construido con bloques disponibles en Scratch que reubica el personaje cuando toca el color magenta. A la derecha, un programa imaginario que realiza lo mismo con el evento (que no existe en Scratch) Al tocar color.

El próximo paso es reconocer que este comportamiento es análogo al de los eventos que provee el entorno, pero **generalizable a cualquier situación que podamos detectar con los sensores**. Además de identificar los bloques necesarios, es valioso escribir la estrategia y hacerlo como parte de un proceso de abstracción y generalización.

De izquierda a derecha: la estrategia general para detectar un evento durante la ejecución del programa; un fragmento de programa que implementa esta estrategia (con el procedimiento Reaccionar al evento); y un programa hipotético en un lenguaje en el que el evento que queremos detectar está disponible con su propio bloque.

Por otro lado, el problema de contar puntos (sumar de a uno, de a tres o incluso de restar) es la motivación para generalizar la noción de contador. Identificamos en qué momentos del desarrollo del proyecto debimos resolver un problema asociado a la variación del puntaje y recuperamos qué

bloques utilizamos para resolverlo, para identificar qué bloques utilizamos y de qué manera lo hicimos.

A continuación se presentan cuatro fragmentos de programa utilizados para contar puntos.

La primera vez que agregamos puntaje, para que el personaje sume un punto al alcanzar el objetivo

Cuando agregamos el segundo personaje y su puntaje por separado.

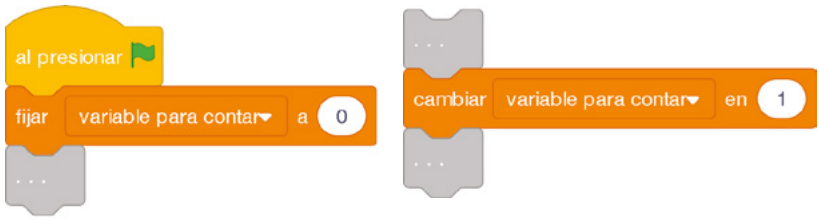
Cuando agregamos el objetivo que suma tres puntos.

Cuando agregamos el objetivo que resta un punto.

A partir de estas soluciones, podemos extraer una estrategia de solución y señalar la importancia de las variables y los bloques fijar y cambiar en su implementación: al comenzar el programa fijamos el valor de la variable que vamos a utilizar en 0 y luego, en el lugar de programa en el que queremos sumar un punto (u otra cantidad), cambiamos el valor de la variable.

Al comenzar:
  Poner valor de la variable en 0

...
  Sumar 1 al valor de la variable
...



Estrategia general de un contador y su implementación en Scratch. Los bloques grises representan fragmentos de programa genéricos.

Para terminar, valoramos el desarrollo incremental de proyectos más complejos como el que hemos emprendido.

¿Cómo fue cambiando el juego después de completar cada consigna? ¿Podrían escribir las reglas de cada una de las versiones obtenidas? ¿Qué relaciones encuentran entre plantear una estrategia para programar el videojuego y la organización del desarrollo en etapas? ¿Qué ventajas tiene construir gradualmente un proyecto?

Nos interesa observar que después de cada paso tenemos una versión del videojuego que puede ser incompleta o distinta de la final, pero que funciona. Tenemos un juego que, aunque no tenga exactamente las mismas reglas que el videojuego que queremos programar, es un juego al que podemos jugar. Este modo de organizar el trabajo facilita la programación porque ordena el proceso y nos permite ir corrigiendo y ajustando a medida que avanzamos (y no recién cuando terminamos de agregar todas las opciones). Podemos relacionar esta idea también con la descomposición en subproblemas: identificamos qué características del juego terminado hay que construir y las vamos haciendo por etapas.

Secuencia didáctica

> Unidad 4:

Repetición condicional

16. **Un videojuego que no sabemos cuándo termina**. Repetición condicional

Secuencia didáctica 16

Un videojuego que no sabemos cuándo termina

Repetición condicional

¿Siempre sabemos cuántas veces hay que repetir una acción? ¿Podemos programar soluciones para recorridos cuya longitud es variable? ¿Cómo hacemos que los objetos en un videojuego se muevan hasta que llegan al borde o alcanzan otro objeto?

Esta secuencia presenta la repetición condicional como una herramienta de programación que permite establecer la repetición de una acción sujeta a una condición de finalización (es decir, repetir hasta que ocurre determinado suceso) en vez de a partir de una cantidad de iteraciones (como la repetición simple). También, propone un recorrido en el que se intercala el trabajo en PilasBloques con la construcción de un videojuego en Scratch. Esta combinación de entornos apunta a integrar en el aprendizaje de esta nueva herramienta una instancia de trabajo en un entorno abierto y sobre un proyecto propio para potenciar su sentido. También, permite presentar problemas en un contexto de programación de un producto más complejo para motivar la continuación del trabajo con PilasBloques.

Actividad 1

Las y los estudiantes resuelven los desafíos de Pilas Bloques *Largos cambiantes*, *Prendiendo las compus* y *Buscando a Guyrá* para aproximarse a la repetición condicional.

Actividad 2

Las y los estudiantes comienzan un proyecto de programación de un videojuego en Scratch en el que el movimiento de los objetos requiere la utilización de la repetición condicional.

Actividad 3

Las y los estudiantes resuelven los desafíos de Pilas Bloques *Fútbol al sur, Luces cambiantes, Laberinto con pelotas* y *Contando astros* que requieren la elaboración de estrategias de solución que combinan repetición condicional con otras herramientas de programación.

Actividad 4

Las y los estudiantes continúan trabajando sobre su videojuego para agregar nuevos comportamientos que requieran la aplicación de la repetición condicional combinada con herramientas conocidas.

Datos curriculares

Nivel: Primaria, segundo ciclo; Secundaria, ciclo básico

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: procedimientos, repetición simple, alternativa condicional, repetición condicional.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad.

Objetivos de aprendizaje

- Identificar características relevantes, comportamientos repetidos y características fijas y variables de una situación problemática claramente enunciada y con límites precisos para elaborar una estrategia de solución a partir de descomponerla en términos de subproblemas que aproveche la repetición condicional.

Saberes previos de CC

Área: Programación

Eje: Lenguajes de programación

- Herramientas de lenguajes de programación: procedimientos, repetición simple, alternativa condicional, sensores, eventos.

Eje: Soluciones a problemas computacionales

- Diseño de soluciones computacionales: estrategia, legibilidad.

Materiales necesarios

- Dispositivos con Scratch 3 instalado o acceso a su versión en línea https://scratch.mit.edu.ar/

Actividad 1
¿Hasta cuándo?

Siguiendo el enfoque por indagación, proponemos desafíos de Pilas Bloques en los que las y los estudiantes deberán analizar un problema para explorar la repetición condicional.

Objetivo >

Se espera que las y los estudiantes se aproximen a la repetición condicional como una herramienta de programación que permite expresar repeticiones que se ejecutan una cantidad variable de veces.

Inicio >

El **propósito de este momento** es presentar situaciones de ejecución de programas familiares para las y los estudiantes (por ejemplo, videojuegos) en las que puedan identificar acciones que se repiten un número indeterminado de veces hasta que ocurre un determinado suceso.

Orientaciones

Comenzamos la actividad ilustrando situaciones en el funcionamiento de programas conocidos por las y los estudiantes en las que exista algún **comportamiento que se repita hasta que pase (o deje de pasar) algo**; para esto podemos mostrar algún videojuego que les sea familiar para analizar cómo se comportan sus distintos elementos.

A partir de este análisis buscamos que las y los estudiantes pongan en palabras la idea de que algunas de las acciones involucradas sobre las que pusimos el foco **se repiten una cantidad de veces *hasta que* sucede algo**.

Para poner en evidencia que, en cada uno de estas situaciones, la repetición sucede hasta que se cumple una condición, podemos escribir una descripción del comportamiento observado en el pizarrón y, en una puesta en común, identificar y remarcar la expresión "hasta que" (u otras similares) como algo que tienen en común todas las situaciones observadas.

Una jugada del videojuego Plants vs. Zombies (UpsDown Studio, 2025).

Por ejemplo, en el juego *Plants vs Zombies* (del cual podemos ver una jugada de ejemplo en ***https://youtu.be/WXK3FGVOmGE***), podemos analizar el movimiento de los soles, los proyectiles y los zombies para identificar, que:

- los soles caen hasta que hacemos clic sobre ellos.
- los proyectiles se mueven hacia la derecha hasta que alcanzan un zombie o salen del escenario.
- los zombies avanzan hasta que son destruidos por los proyectiles.
- los zombies se mueven hacia la izquierda hasta que encuentran un obstáculo o una planta.

Para aproximarnos a la idea de repetición condicional, podemos identificar cuáles son las acciones que están sucediendo bajo esas condiciones. Tanto los soles, los zombies y los proyectiles realizan la acción de avanzar.

¿Es posible usar repeticiones simples para resolver estas situaciones? ¿Por qué?

Hacemos la pregunta para diferenciar estos comportamientos de aquellos que resolvían con repeticiones simples. La clave para responder la pregunta es observar que la cantidad de veces que se deben repetir las instrucciones no puede ser definida, sino que debe variar en función de la ejecución. Por ejemplo, en *Plants vs. Zombies*, la cantidad de veces que los proyectiles lanzados por una planta deberán repetir la acción de avanzar para llegar al borde derecho depende de dónde el jugador ubique a la planta en el escenario; la cantidad de veces que avanzan los zombies depende de cuántas veces sean impactados por los proyectiles durante la ejecución, etcétera.

Desarrollo >

El **propósito de este momento** es introducir a las y los estudiantes a la repetición condicional a partir de desafíos de programación.

Orientaciones

Invitamos a las y los estudiantes a formar grupos heterógeneos[1] para resolver el desafío de Pilas Bloques ***Largos cambiantes***[2].

Para comenzar, habilitamos un momento de exploración en el que puedan ensayar soluciones para reconocer las limitaciones de las herramientas conocidas e investigar si existen otras nuevas que les permitan resolver el desafío.

Algunos de los escenarios posibles del desafío *Largos cambiantes*.

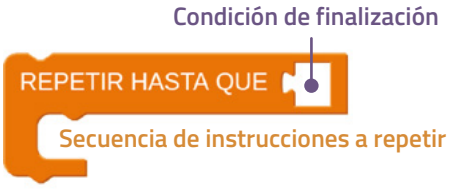
Cuando lo consideremos pertinente, hacemos una puesta en común para relevar los descubrimientos de los grupos.

¿Qué características tienen los escenarios del desafío presentado? ¿En qué se parece a los que ya conocían? ¿Cuál es la diferencia? ¿Pudieron identificar características que se mantienen en todos los tableros y otras que cambien entre uno y otro? ¿Pudieron armar una solución que funcione para todos los tableros posibles? ¿Exploraron los nuevos bloques? ¿Podrían explicar cómo funcionan?

Buscamos con las respuestas que logren reconocer que el desafío abordado tiene un parecido con los vistos en otras secuencias didácticas de esta colección: se trata de un recorrido en el que debemos realizar la misma acción en todos los casilleros, sin embargo, en este desafío, la longitud de los recorridos varía entre un escenario y otro. Entonces, para

[1] Se pueden consultar dinámicas lúdicas para el armado de grupos heterogéneos en "Enseñar computación desde la mirada de la Educación Sexual Integral (ESI)", 10.
[2] Disponible en "Nivel intermedio", "Repetición condicional", del sitio ***https://pilasbloques.program.ar/online/***

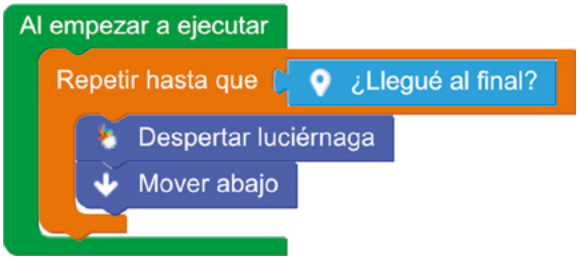
resolverlo con una única solución que atienda a todos los escenarios posibles, hay que prever que **la cantidad de repeticiones a realizar va a variar en cada escenario** (que es una situación similar a la de los desafíos de alternativa condicional). Contamos con una nueva herramienta de programación para atender este requerimiento: el bloque Repetir hasta que... Este bloque permite indicar que una acción debe repetirse, pero, en vez de indicar la cantidad de repeticiones, indicamos la condición que finaliza la repetición. Por eso, decimos que es una **repetición condicional**.



El bloque de repetición condicional y sus componentes.

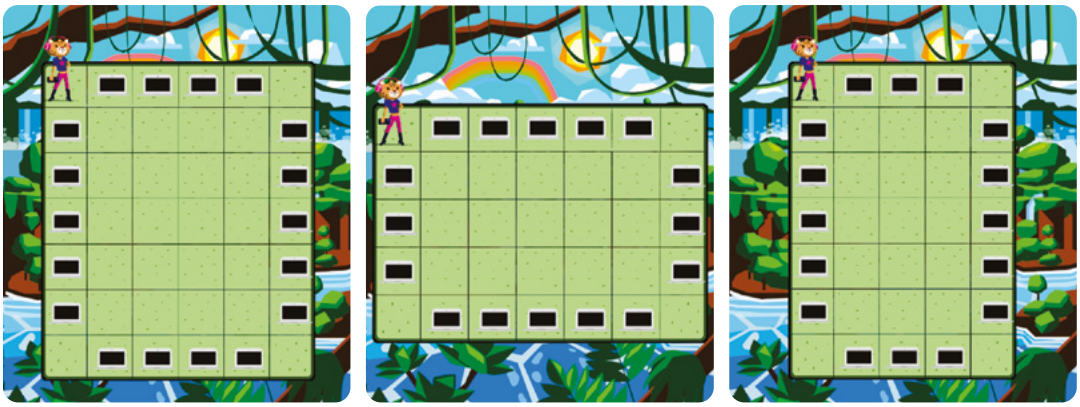
A continuación de la explicación, habilitamos una nueva instancia de programación para que las y los estudiantes que aún no lo hayan hecho incorporen la nueva herramienta a su programa.

Cuando todos los grupos hayan resuelto el desafío, hacemos una puesta en común para que cada grupo comparta su solución. Observamos entre todas y todos que, para resolver el desafío, es necesario combinar el bloque Repetir hasta que... con el sensor ¿Llegué al final? para que Ivoty se mueva hasta llegar al final del recorrido, sin importar la longitud del tablero, repitiendo las instrucciones Despertar luciérnaga y Mover abajo.



Una solución posible al desafío Largos cambiantes, que requiere el uso de la repetición condicional.

A continuación, les proponemos resolver el desafío de Pilas Bloques ***Prendiendo las compus*** con el propósito de reforzar el uso de repeticiones condicionales para cumplir el objetivo.

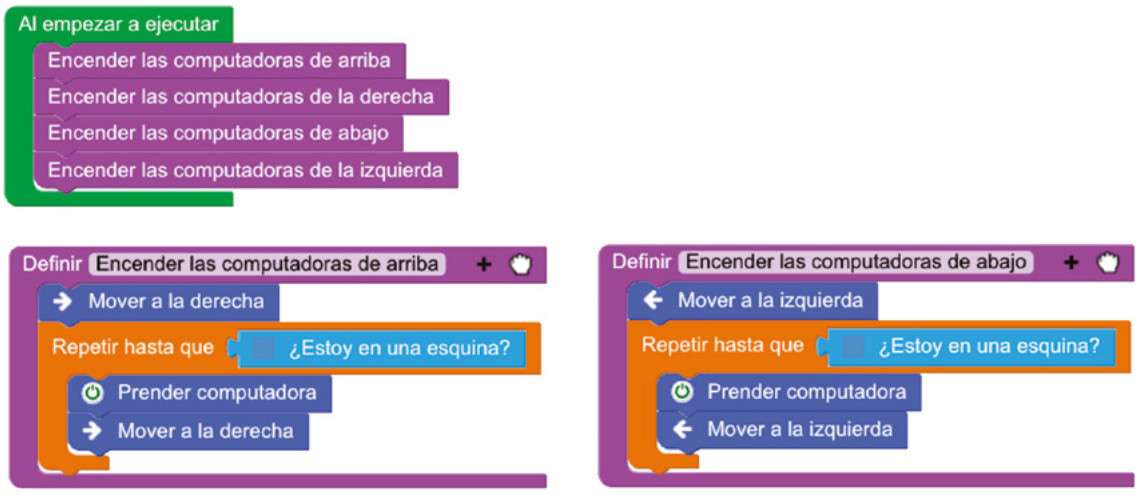


Tres escenarios posibles del desafío *Prendiendo las compus*.

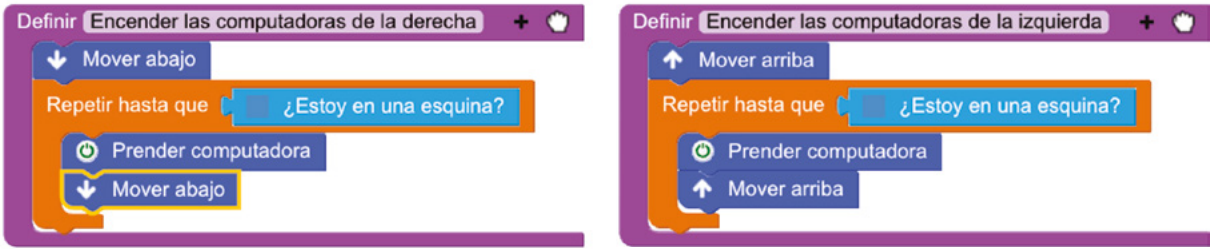
Para los grupos que necesiten ayuda durante la exploración, podemos alentarlos a observar los distintos escenarios para identificar características constantes (la posición de Ivoty y la disposición de las computadoras: están en todos los casilleros del borde del tablero, excepto en las esquinas) y variables (el tamaño del tablero y, por ende, la longitud de las filas y las columnas con computadoras).

Como siempre, alentamos a elaborar una estrategia para guiar la programación y, en este caso, poniendo especial atención a las variaciones de los tableros.

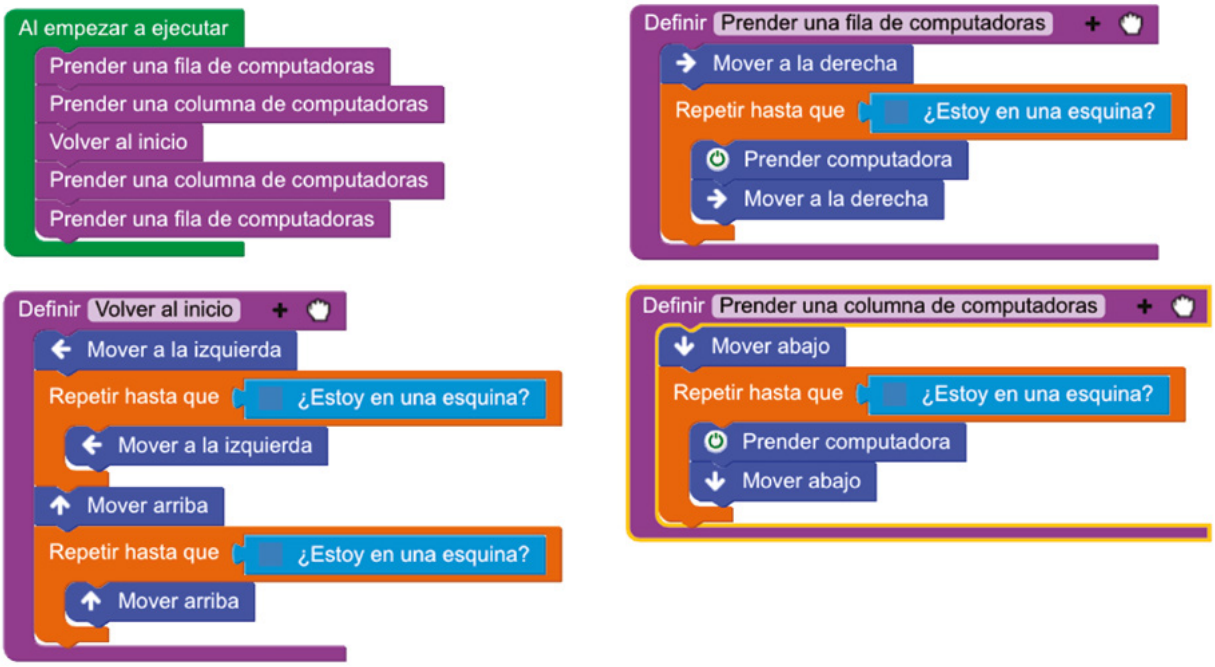
Una estrategia posible para resolver el desafío es recorrer todo el borde del tablero en sentido horario. Otra estrategia posible es contar con tres procedimientos: uno para los bordes horizontales, otro para los verticales y otro para volver al casillero inicial. En ambos casos, se aprovecha la estructura del tablero para definir los procedimientos y se utiliza la repetición condicional con el sensor ¿Estoy en una esquina? para resolver cada hilera.



Una solución posible que recorre el borde del tablero en sentido horario en el desafío *Prendiendo las compus*.



Una solución posible que recorre el borde del tablero en sentido horario en el desafío *Prendiendo las compus*.



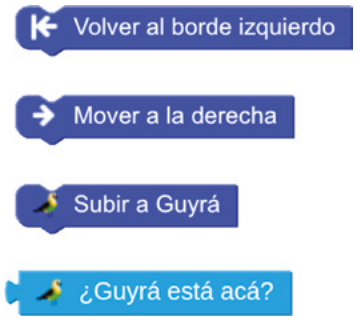
Una solución que reutiliza los procedimientos para resolver las hileras horizontales y verticales en el desafío *Prendiendo las compus*.

Luego de observar las soluciones elaboradas por las y los estudiantes, hacemos una puesta en común para compartir y comparar las estrategias y las soluciones posibles, señalando especialmente cómo se resolvieron las variaciones en el tablero y el rol de la repetición condicional en la solución.

Cerramos con el desafío ***Buscando a Guyrá***. Invitamos a los grupos a que lo exploren y comiencen a resolverlo.

Tres escenarios posibles del desafío con distintas posiciones de los personajes.

En función del nivel de avance o de la necesidad de acompañamiento de los grupos, podemos hacer una puesta en común para compartir una característica clave del problema: en algunos escenarios, Guyrá se encuentra a la izquierda de Capy y, en otros, a la derecha. También es posible identificar observando los bloques disponibles que Capy solo puede moverse a la derecha y que no hay sensores disponibles que nos brinden información sobre hacia qué lado se encuentra Guyrá, lo que limita fuertemente las estrategias de solución.



Primitivas y sensores disponibles del desafío *Buscando a Guyrá*.

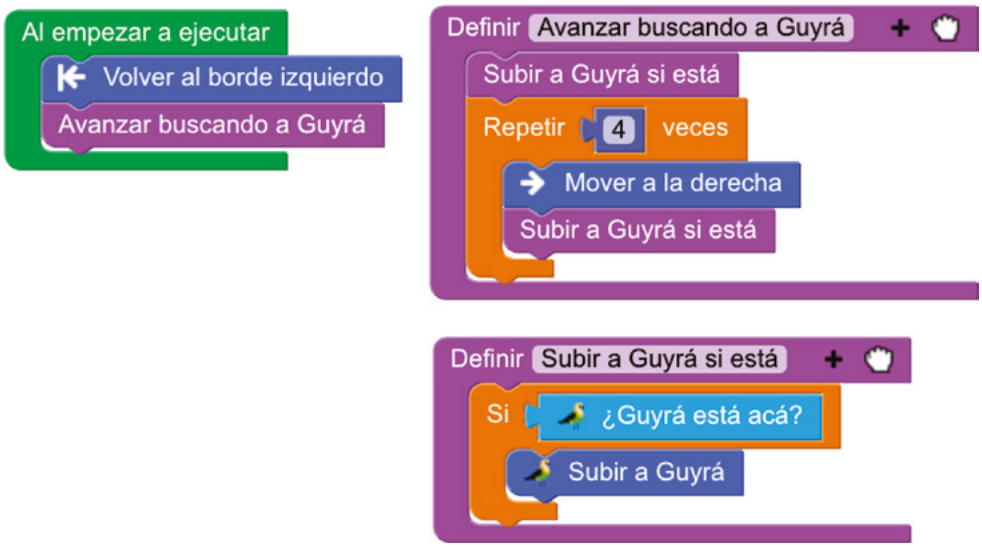
Para los grupos que necesiten asistencia adicional, podemos reforzar el análisis del problema y las limitaciones que imponen los bloques para que puedan concluir que primero deben llevar a Capy al extremo izquierdo y luego avanzar hacia la derecha **hasta que** se encuentre con Guyrá.

Cuando todos los grupos hayan resuelto el desafío, invitamos a compartir las estrategias y sus implementaciones. Al igual que en los casos anteriores, hacemos especial énfasis en cómo hicieron uso de la repetición condicional. En las soluciones, podemos identificar tres partes: primero, avanzar hasta el borde izquierdo para asegurarnos de que Guyrá está a la derecha de Capy; a continuación, avanzar hacia la derecha hasta encontrar a Guyrá; y, finalmente, subir a Guyrá.

Una solución posible con repetición condicional del desafío *Buscando a Guyrá*.

Sin embargo, este desafío también se puede resolver con repetición simple sin utilizar la repetición condicional. En el caso de que algún grupo haya propuesto esta solución, alentamos a que la compartan con el resto de la clase para compararla con las que sí lo hacen. Podemos observar que se trata de una estrategia de recorrido con caso borde

y elementos cambiantes en las casillas (como, por ejemplo, la que usaron para resolver el desafío ***Alineando telescopios*** en la secuencia *Resolvemos recorridos cambiantes. Alternativa condicional y repetición*). Sin embargo, alentamos a observar la ejecución del programa en varios casos, para identificar que, independientemente de la posición de Guyrá, Capy siempre recorre todos los casilleros. En este sentido, a partir de la comparación de las soluciones, es posible observar que la repetición condicional permite hacer una solución que termine la ejecución antes.



Una solución al desafío con repetición simple del desafío *Buscando a Guyrá*.

Cierre >

El **propósito de este momento** es recuperar el trabajo realizado durante la actividad para reforzar la conceptualización de la repetición condicional.

Orientaciones

Habilitamos una puesta en común y un espacio de reflexión en voz alta como una instancia de metacognición para identificar que conocimos una nueva herramienta de programación (la repetición condicional), qué características de los desafíos hicieron que fuera necesario utilizar esta herramienta, cómo funciona la herramienta y por qué nos permitió resolver estos desafíos.

Para reforzar la conceptualización de la repetición condicional, podemos contrastarla con la repetición simple preguntando por qué no fue suficiente con la que ya conocíamos.

¿Qué nueva herramienta de programación conocieron en esta actividad? ¿Cuándo la conocieron? ¿Cómo se dieron cuenta de que necesitaban una nueva herramienta y no alcanzaba con lo que ya conocían?

En el primer desafío (***Largos cambiantes***), se introdujo la repetición condicional. Para resolver el desafío, fue necesario repetir un conjunto de acciones (despertar la luciérnaga y avanzar hacia abajo) una cantidad de veces que no se sabía cuál iba a ser, pues el largo del recorrido cambiaba entre una ejecución y otra. Esta característica del problema impide usar la repetición simple para resolverlo, porque la cantidad de veces a repetir las acciones varía en cada escenario posible y, a partir de esa limitación exploraron el funcionamiento del nuevo bloque.

¿Cómo explicarían el funcionamiento de la repetición condicional? ¿Por qué fue útil para resolver los tres desafíos de la actividad? ¿Cómo la aplicaron en cada caso?

Este tipo de repetición repite una secuencia de acciones hasta que se cumple una condición, y para saber si se ha cumplido, se utilizan sensores. En el primer desafío, la solución requiere el uso del sensor ¿Llegué al final? para expresar la idea de avanzar hasta el final sin tener que conocer cuántos casilleros hay que moverse.

En el segundo desafío (***Prendiendo las compus***), era necesario recorrer los bordes del escenario, pero su extensión variaba entre una ejecución y otra. Para eso, utilizamos la repetición condicional con el sensor ¿Estoy en una equina?.

En el tercer desafío (***Buscando a Guyrá***), había que avanzar hasta Guyrá, pero como no era posible conocer su ubicación de antemano, no era posible definir en el programa una cantidad de veces para avanzar. Para elaborar una solución que no recorriera casilleros de más, aprovechamos la repetición condicional y el sensor ¿Guyrá está acá?.

¿Se les ocurre cómo se podría utilizar la repetición condicional para programar los comportamientos que identificaron al inicio?

Para profundizar la conceptualización y motivar la generalización proponemos elaborar estrategias o programas hipotéticos para los comportamientos que identificaron en el inicio de esta actividad.

Repetir hasta que ¿No me quedan más vidas?
    Avanzar a la izquierda

Repetir hasta que ¿Toqué un zombie o salí del escenario?
    Avanzar a la derecha

Programas hipotéticos para controlar a los zombies (izq.) y a los proyectiles (der.) en *Plants vs. Zombies*.

Actividad 2
Un videojuego con repetición condicional

Esta actividad se propone como una instancia de trabajo en un entorno abierto en la que las y los estudiantes recuperen sus aprendizajes en Pilas Bloques para la construcción de un videojuego (pequeño) propio en otro entorno de programación (Scratch).

Objetivo >

Se espera que las y los estudiantes:

- Reconozcan cuándo es necesario usar la repetición condicional en un nuevo entorno de programación y para un problema diferente.
- Utilicen la repetición condicional en un nuevo entorno de programación y para un problema diferente.
- Comiencen a combinar la repetición condicional con otras herramientas de programación conocidas.

Inicio >

El **propósito de este momento** es recuperar la noción de la repetición condicional vista en el entorno Pilas Bloques y promover su reconocimiento en un entorno de programación distinto.

Comenzamos la actividad presentando el videojuego completo y analizamos su comportamiento, prestando especial atención al movimiento de los objetos. Pueden encontrar una versión del proyecto completo en el archivo Andrómeda_completo.sb3 que acompaña esta secuencia.

Contamos que, al igual que en otros proyectos de programación, presentaremos una serie de desafíos a partir de los cuales iremos completando gradualmente nuestro programa.

305

Desarrollo >

El **propósito de este momento** es que las y los estudiantes analicen el movimiento de los objetos para identificar comportamientos que involucran la repetición condicional y propongan estrategias de solución que utilicen esta herrmienta con y sin combinarla con otras herramientas de programación.

Orientaciones

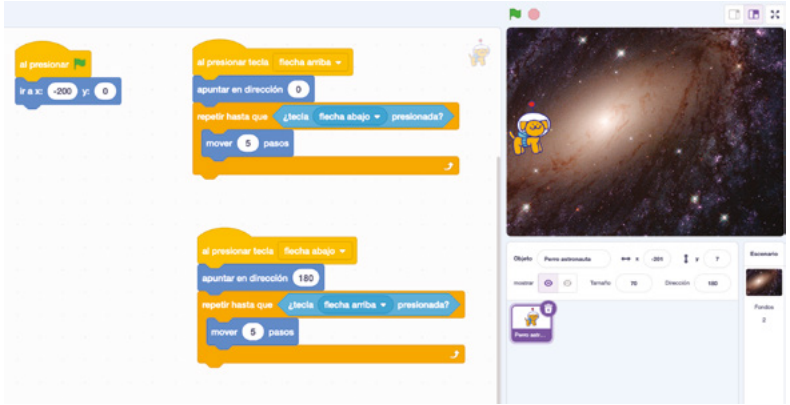
Para comenzar a trabajar, comenzamos un nuevo proyecto en Scratch con un personaje y presentamos el primer desafío de programación.

> ***Desafío 1.*** Programar el movimiento del personaje para que se mueva a lo largo del borde izquierdo de esta manera: comienza a moverse en una dirección al presionar una tecla y cambia al presionar la contraria.

Alentamos el análisis del movimiento del personaje en el videojuego completo para reconocer analogías entre este comportamiento y los desafíos de Pilas Bloques que abordamos en la **Actividad 1**. En ambos casos hay **acciones que deben repetirse y no podemos saber cuántas veces** de antemano, pero **sí podemos identificar cuándo deben dejar de repetirse**. Asociamos este comportamiento con la **repetición condicional**.

Para quienes lo necesiten, podemos pedirles que describan el movimiento del personaje para identificar que hay un movimiento que comienza al presionar una tecla y se repite **hasta que** se presiona la tecla contraria.

También podemos alentarlos a revisar los bloques de la categoría Control y Sensores para que encuentren los bloques necesarios (repetir hasta que y ¿tecla ( ) presionada?).



Una solución posible para el movimiento del personaje.

¿Qué estrategia utilizaron para resolver este desafío? ¿Qué bloques utilizaron para programarla? ¿Qué parecidos encuentran con el trabajo en Pilas Bloques?

En una breve puesta en común compartimos las estrategias y señalamos el uso de la repetición condicional, reforzando la importancia de **transferir las herramientas de programación entre los distintos entornos**.
Podemos recuperar las soluciones a los desafíos de la **Actividad 1** para compararlas con las de este desafío.

La repetición condicional es una herramienta común a muchos lenguajes de programación, hay bloques para esto tanto en Scratch (izq.) como en Pilas Bloques (der.).

Presentamos el próximo desafío, que apunta a agregar enemigos u obstáculos.

Desafío 2. Incorporar un obstáculo o enemigo que se mueva como los meteoritos del juego terminado.

Alentamos a los grupos a analizar el comportamiento de los meteoritos en el juego terminado que brindamos como ejemplo, para describirlo y pensar una estrategia de solución.

¿Dónde aparece el meteorito? ¿Hacia dónde se mueve? ¿Empieza su movimiento siempre en el mismo lugar? ¿Cuándo termina de moverse? ¿Qué puede pasar cuando termina de moverse?

A quienes lo necesiten, guiamos con preguntas. Articulamos algunas instancias de análisis y reflexión dentro de los grupos con puestas en común o reflexiones más guiadas entre todas y todos, dependiendo de la necesidad de acompañamiento.

Nos interesa que cada grupo identifique que este objeto:

- Aparece cerca del borde derecho.
- Se mueve hacia la izquierda.

- Cuando toca al personaje, termina el juego.
- Cuando alcanza el borde izquierdo, vuelve a empezar.

Después de la programación, hacemos una puesta en común para que los grupos compartan sus estrategias, señalando particularmente dónde utilizaron la repetición condicional y por qué (qué parte del problema identificaron como algo que se repite hasta que pasa algo).

```
Al comenzar:
  Repetir hasta que ¿Tocando al personaje?:
  |  Mover un paso a la izquierda
  |  Si ¿Tocando borde izquierdo?:
  |  |  Reubicar cerca del borde derecho
  |  |
  |
  Decir "Perdiste"
  Detener todo.
```

```
Al comenzar:
  Repetir por siempre:
  |  Avanzar a la izquierda y perder si toca el personaje.
  |  Reubicar cerca del borde derecho
  |

Avanzar a la izquierda y perder si toca el personaje:
  Repetir hasta que ¿Tocando el borde izquierdo?:
  |  Mover un paso a la izquierda
  |  Si ¿Tocando al personaje?:
  |  |  Decir Perdiste.
  |  |  Detener todo.
  |  |
  |
```

Dos estrategias posibles para resolver el desafío, con énfasis en la división en subproblemas.

Material de referencia para docentes

Hay otras características que hacen más interesante el juego al volverlo menos predecible. Podemos proponerlas como continuaciones posibles, teniendo en cuenta cómo debería modificarse la estrategia propuesta para el **Desafío 2**.

- El meteorito aparece después de un tiempo de que empieza el juego.
- Pasa un tiempo aleatorio entre que el meteorito llega al borde y vuelve a aparecer.
- El meteorito no aparece siempre en la misma posición. Cada vez que aparece:
  - » varía un poco la posición horizontal, pero siempre está cerca del borde derecho
  - » lo hace en una posición vertical distinta (no sabemos "a qué altura" va a aparecer).

Para resolver estos aspectos puede ser necesario recordar los bloques de espera, para esconder o mostrar objetos y de números aleatorios. Para implementarlos alcanza con agregar una espera al comienzo del programa del meteorito (que puede ser una cantidad fija o una cantidad aleatoria) y comportamientos aleatorios en "Reubicar cerca del borde derecho".

Cierre >

El **propósito de este momento** es abstraer las estrategias para identificar el uso de la repetición condicional combinada con otras herramientas de programación.

Orientaciones

Invitamos a los grupos a que cuenten cómo quedaron sus estrategias y programas y, en particular, cómo utilizaron la repetición condicional y cómo la combinaron con otras herramientas. Podemos ir tomando nota para identificar algunos esquemas generales.

Es posible una solución que no utilice bloques de repetición condicional (usando solo repetición por siempre y alternativas condicionales para interrumpir o reiniciar). De todas maneras, podemos identificar que existen comportamientos que se *repiten hasta* que ocurre un suceso. Por ese motivo, y por motivos análogos a por los que alentamos el uso de procedimientos, nos parece positivo utilizar los bloques de repetición condicional: se comunica mejor la estrategia, es más claro identificar subproblemas y mejora la legibilidad del programa.

Una conclusión instrumental, pero útil, es la importancia de la repetición condicional para controlar el movimiento de los objetos en los videojuegos. Muchas veces los objetos o personajes se mueven "hasta que" pasa algo (chocan, pasa un tiempo, alcanzan cierto punto, tocan el borde, etc.). Podemos recuperar los comportamientos que habíamos analizado en el inicio de la **Actividad 1**. Por lo tanto, estos bloques serán frecuentes cuando programemos videojuegos con objetos que se mueven.

También explicitamos la presencia de la repetición condicional y la repetición condicional combinada con otras herramientas como la alternativa condicional y la repetición por siempre. A medida que la complejidad de la solución aumenta (por ejemplo, porque tenemos

más herramientas combinadas entre sí), se vuelve más importante la instancia de análisis del problema para contar con una estrategia clara que ordene la programación. Por otro lado, combinar estas herramientas amplían las posibilidades de los comportamientos que podemos programar (y por ende, que pueden tener los objetos de nuestros videojuegos). Esta es la motivación para la actividad siguiente.

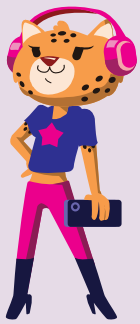
Actividad 3
Repetición condicional y mucho más

En esta actividad, las y los estudiantes resuelven los desafíos de Pilas Bloques ***Fútbol al sur, Luces cambiantes***, ***Laberinto con pelotas*** y ***Contando astros*** para lo que deberán explorar la combinación de la repetición condicional con otras herramientas de programación presentadas en secuencias anteriores.

Objetivo >

Se espera que las y los estudiantes:

- Analicen problemas de programación, teniendo en cuenta qué características del escenario permanecen fijas y cuáles varían, para reconocer la necesidad de utilizar y combinar los procedimientos, la repetición simple, la alternativa condicional y la repetición condicional.
- Diseñen e implementen estrategias, a partir de la división en subproblemas, que combinen la repetición condicional con otras herramientas de programación conocidas.



Inicio >

El **propósito de este momento** es motivar la combinación de la repetición condicional con otras herramientas de programación como una manera de expresar nuevos tipos de comportamientos a partir de las estrategias elaboradas en la **Actividad 2** para el movimiento de los objetos del videojuego.

Invitamos a las y los estudiantes a recuperar los comportamientos que programaron en la **Actividad 2** y las estrategias que propusieron, para identificar en ellas cómo aparece combinada la repetición condicional con otras herramientas conocidas. Podemos recuperar las conclusiones del cierre en este sentido también.

Desarrollo >

El **propósito de este momento** es proponer desafíos que requieran la elaboración de estrategias que combinen repetición condicional con otras herramientas.

Orientaciones

Presentamos el desafío de Pilas Bloques ***Fútbol al sur*** e invitamos a los grupos a observar los diversos escenarios del desafío para identificar aspectos que varían y otros que permanecen constantes. Este análisis es el insumo para la elaboración de la estrategia de solución.

Tres escenarios posibles del desafío *Fútbol al sur*: el largo de cada fila varía, en cambio, la cantidad de filas y la presencia de una pelota al final de cada una son constantes.

Avanzan con la programación y el refinamiento de la estrategia. Recorremos los grupos y si fuera necesario, acompañamos el análisis de los escenarios y la identificación de los aspectos que permanecen constantes (8 filas que empiezan con un casillero azul y terminan con una pelota) y los que varían (el largo de cada fila).

Si lo consideramos pertinente, en función de las dificultades que observemos en los grupos, podemos hacer una puesta en común para identificar que la estrategia requiere la combinación de la repetición simple para resolver las 8 filas con la repetición condicional para resolver cada fila. En este sentido, también reforzamos la importancia de definir procedimientos para aislar estos subproblemas. Un indicador de que propusimos una buena separación en subproblemas es que la solución no tenga bloques de repetición andidados.

Una solución posible para el desafío *Fútbol al sur*.

Cuando hayan resuelto el desafío, hacemos una puesta en común para que los grupos presenten los programas y las estrategias.

¿Cómo utilizaron cada tipo de repetición? ¿Cómo eligieron cuál usar? ¿Qué subproblemas identificaron (y qué procedimientos definieron en consecuencia)?

A continuación, avanzamos con el desafío ***Luces cambiantes***: invitamos a los grupos a resolverlo, prestando especial atención a los diversos escenarios que aparecen en cada ejecución.

Tres escenarios posibles del desafío *Luces cambiantes*.

Proponemos a los grupos que avancen en el análisis de los escenarios, la elaboración de la estrategia y la programación del desafío. Como antes, alentamos la identificación de características fijas y variables entre los distintos escenarios. También podemos alentar la identificación de desafíos

similares y recuperar sus soluciones como punto de partida para resolver este (por ejemplo, ***Largos cambiantes*** de la **Actividad 1** o ***Tomando buenas fotos*** de la secuencia "Resolvemos recorridos cambiantes").

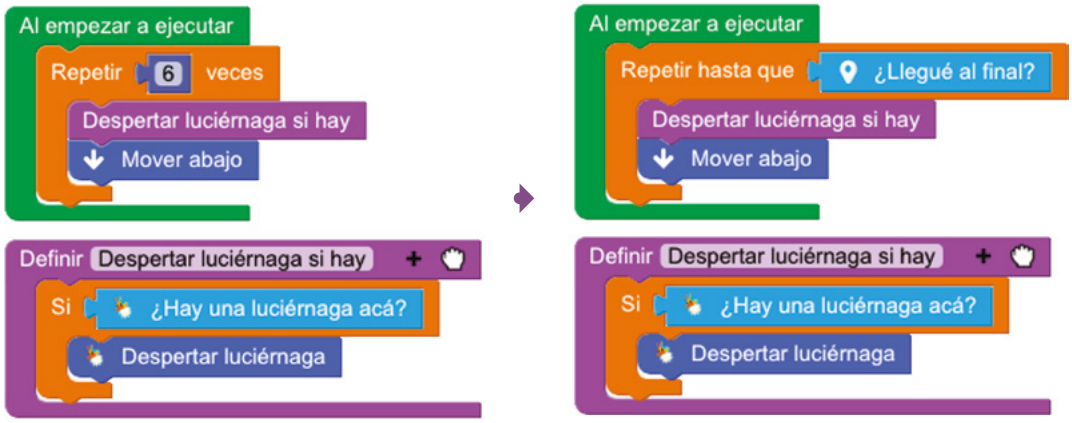
Cuando todos hayan terminado, hacemos una puesta en común para compartir el trabajo y las reflexiones que ocurrieron durante el desarrollo.

Compartimos las estrategias y los programas para identificar que está involucrada la repetición condicional (para resolver el subproblema del largo variable del escenario) junto con la alternativa condicional (para resolver el subproblema de que cada casillero puede o no contener una luciérnaga).

Si recurrieron a otros desafíos, los recuperamos para relacionar las diferencias que identificaron con las modificaciones que tuvieron que hacer y las herramientas de programación que utilizaron para esto.

Por ejemplo, podemos pensar que este desafío es como ***Tomando buenas fotos***, pero con el largo del recorrido variable. Por ese motivo, reemplazamos la repetición simple en la solución por una repetición condicional.

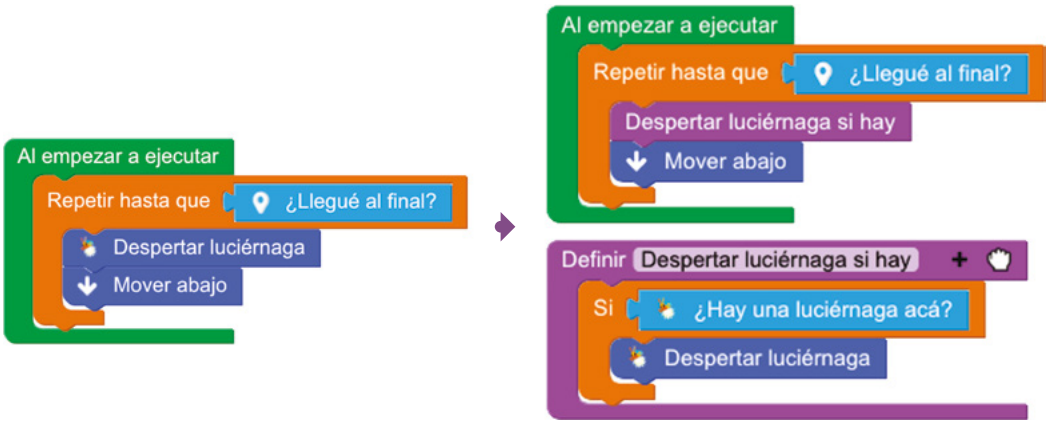
Escenarios posibles de *Tomando buenas fotos*. Varían qué casillas contienen luciérnagas pero el largo del recorrido es constante.



Soluciones para *Tomando buenas fotos* (izq.) y *Luces cambiantes* (der.). Comparando las soluciones, vemos que la única diferencia es el tipo de repetición utilizado.

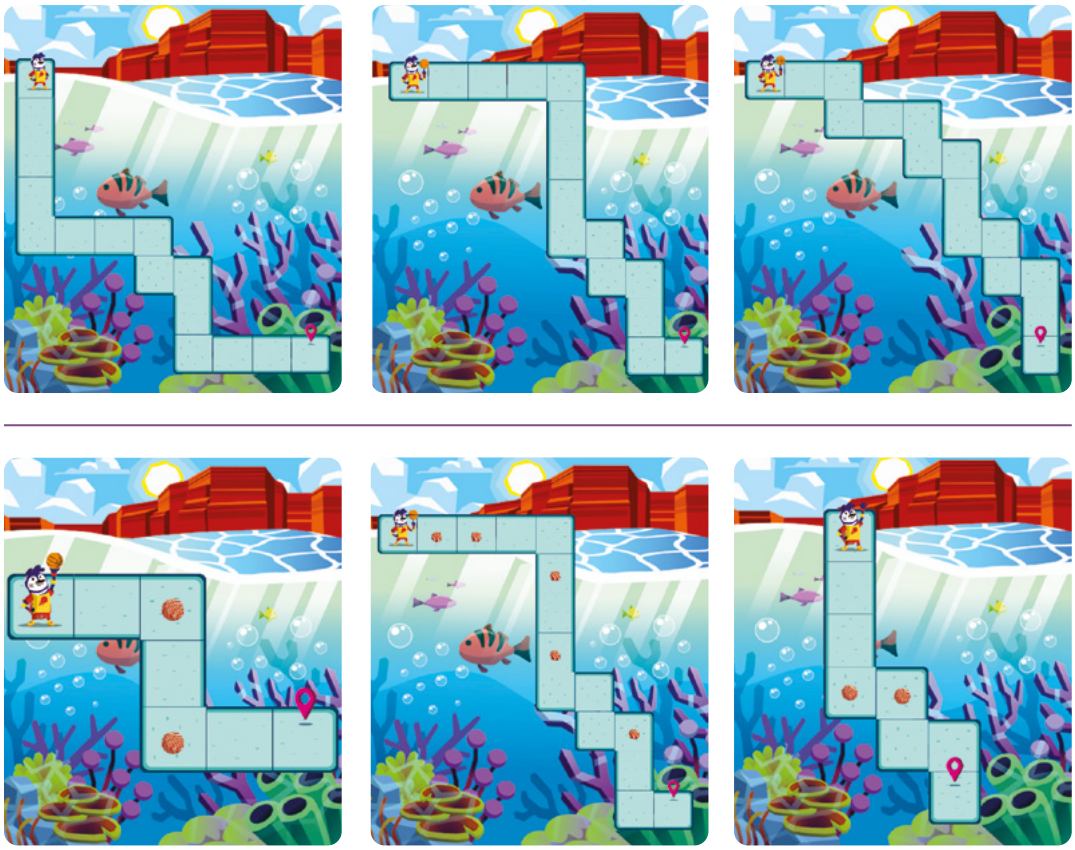
También podemos comparar este desafío con ***Largos cambiantes***. En ambos desafíos el largo del recorrido es variable (y por eso conservamos el uso de la repetición condicional), pero, en este, no todos los casilleros contienen luciérnagas y, por eso, agregamos una alternativa condicional en la resolución de cada casillero. Para resolver el segundo desafío se debe agregar la alternativa condicional alrededor de la primitiva Despertar luciérnaga. Además, se puede definir un procedimiento para explicitar el subproblema y conservar una estructura similar en el programa principal (una repetición con una acción y un avance dentro).

Escenarios posibles de *Largos cambiantes*: el largo del recorrido es variable, pero todos los casilleros contienen luciérnagas.



Comparación de soluciones de los desafíos *Largos cambiantes* (izq.) y *Luces cambiantes* (der.).

El próximo desafío de Pilas Bloques a resolver es ***Laberinto con pelotas***. Alentamos a explorar los distintos escenarios del desafío y similitudes con otros desafíos que hayan resuelto previamente, por ejemplo, ***Barrilete Cósmico*** (resuelto en la secuencia didáctica de esta colección "Programamos estrategias para problemas cambiantes").

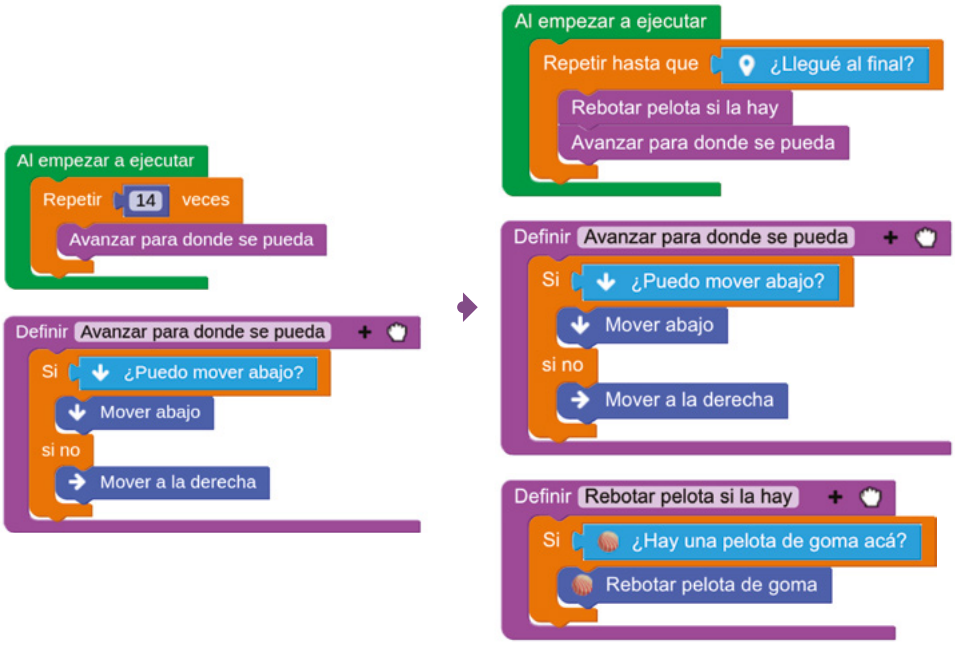


Comparamos escenarios posibles del desafío *Barrilete cósmico* (arriba) con los de *Laberinto con pelotas* (abajo).

Si comparamos los desafíos, podemos identificar que:

- La disposición de los casilleros sigue el mismo criterio en ambos desafíos: el próximo casillero siempre aparece a la derecha o abajo del anterior.
- El largo del recorrido en ***Barrilete cósmico*** es constante, pero en ***Laberinto con pelotas*** varía.
- En ***Barrilete cósmico*** no había nada para hacer en cada casillero. En este desafío, en los casilleros pueden aparecer pelotas que Chuy debe rebotar cuando las encuentra.

Invitamos a resolver este desafío. Quienes dispongan de la solución para ***Barrilete cósmico*** o ***Jugadore de toda la cancha*** (resuelto en la misma secuencia didáctica) pueden abrirla para utilizarla como punto de partida. Mientras continúan con la programación, recorremos los grupos para identificar dificultades en la incorporación de las modificaciones que vimos. Recordamos la importancia de definir procedimientos para separar subproblemas y conseguir programas más legibles.



Solución de *Barrilete cósmico* (izq.) y *Laberinto con pelotas* (der.): se reemplaza la repetición simple por la repetición condicional y se agrega el procedimiento Rebotar pelota si la hay.

A modo de puesta en común, compartimos estrategias y soluciones. Nos interesa especialmente cómo combinaron las distintas herramientas de programación y si utilizaron soluciones a otros desafíos, cuáles, por qué, y qué decidieron modificar, cómo y por qué.

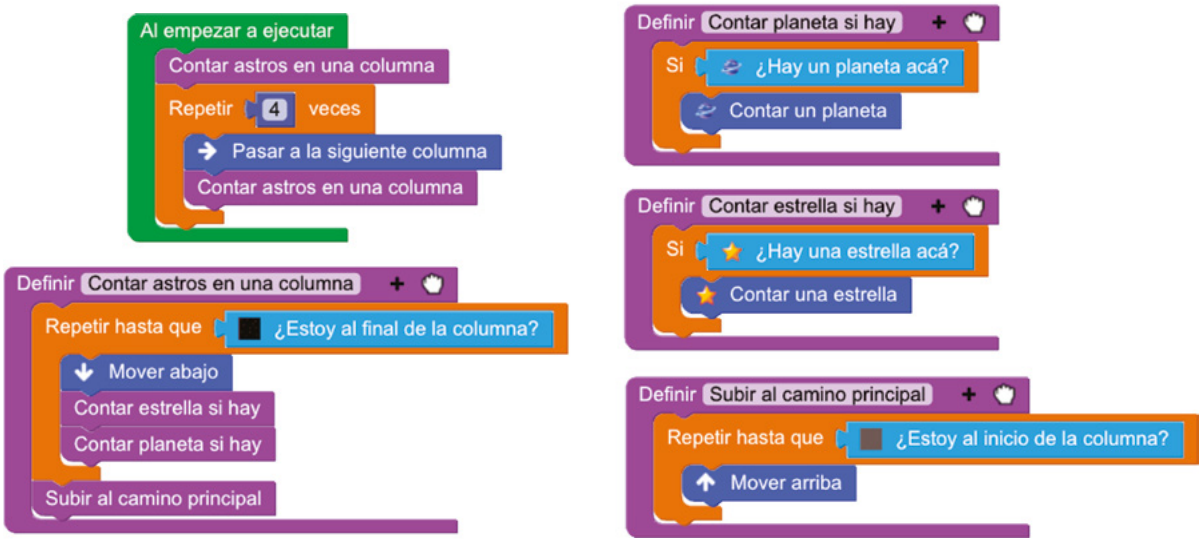
Para terminar esta actividad, proponemos un último desafío de Pilas Bloques en el que aparecen combinadas todas las herramientas de programación que abordamos en la colección.

Presentamos el desafío ***Contando astros*** y brindamos un espacio de trabajo autónomo para que lo resuelvan. Es importante que como parte de este trabajo analicen los distintos escenarios del desafío (para identificar características constantes y variables) y planteen un esbozo de estrategia de solución en el que se diferencien claramente los subproblemas y el uso de las distintas herramientas de programación para cada uno.

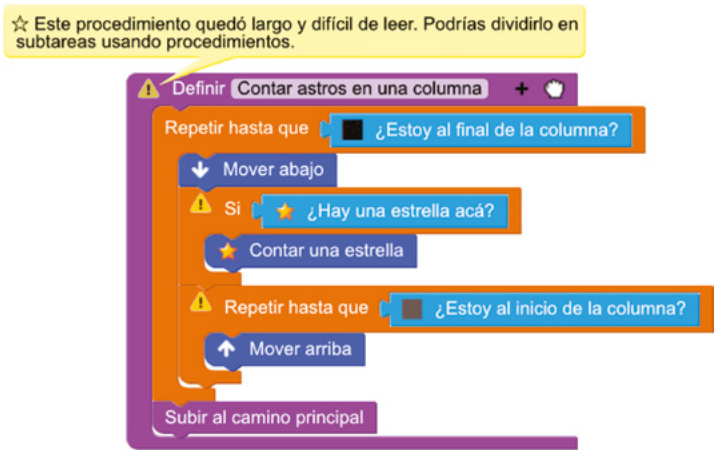
Algunos escenarios posibles del desafío *Contando astros*.

Brindamos asistencia a quienes lo necesiten y, cuando esté concluido el desafío, ponemos en común el análisis del problema y, como consecuencia, las estrategias de solución y las herramientas de programación utilizadas.

A partir del análisis, esperamos que identifiquen que hay una cantidad fija de columnas (5) y que todas siguen un mismo criterio: el largo es variable y el contenido de los casilleros también (un planeta, una estrella o nada). Por lo tanto, una estrategia de solución puede ser repetir cinco veces la solución de una columna, y para esto podemos utilizar la repetición simple (teniendo en cuenta el caso de borde). Para resolver cada columna utilizamos una repetición condicional para avanzar hasta el final (independientemente del largo de la columna) y en cada casillero, verificamos para cada tipo de astro, si está presente (mediante una alternativa condicional con el sensor correspondiente) y, en caso de que sí, lo contamos. Para volver al inicio de la columna, utilizamos nuevamente una repetición condicional dado que el largo es variable. Reforzamos en este proceso la definición de procedimientos para separar los subproblemas. Si estuvieran activadas las sugerencias, es posible que, de no hacerlo, salga una sugerencia en este sentido.



Una solución posible con procedimientos para atender a cada subproblema del desafío *Contando astros*.



Procedimiento muy extenso y el aviso de Pilas Bloques para que se divida en otros procedimientos.

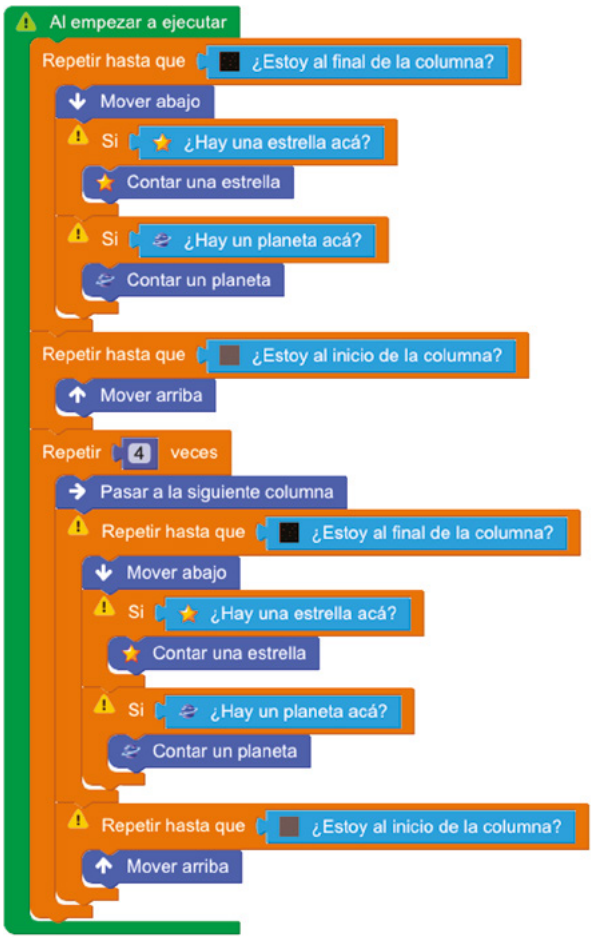
Cierre >

El **propósito de este momento** es explicitar el análisis que hicieron de cada desafío resuelto y las decisiones de programación que tomaron en función de ello (en particular, qué herramienta de programación utilizaron) para reforzar la conceptualización de estas herramientas y su combinación.

Orientaciones

Repasamos las soluciones a los desafíos resueltos en esta actividad para identificar algunos subproblemas cuya solución esté claramente asociada a una herramienta de programación. Por ejemplo, en el desafío ***Fútbol al sur***, podemos identificar como un subproblema cada fila que se resuelve utilizando repetición condicional y, como otro, la repetición de esta solución para todas las filas (que se resuelve utilizando repetición simple). Al identificar estos subproblemas también surgirá la necesidad de combinar estas herramientas y hacerlo de la manera más clara posible, para lo que será importante la definición de procedimientos.

Por ejemplo, en el último desafío, identificamos numerosos subproblemas asociados a distintas herramientas de programación y en distintos niveles. Identificamos: la repetición de cada columna, asociada a la repetición simple; el recorrido de cada columna contando astros, asociado a la repetición condicional (que incluía la identificación de qué tipo de astro estaba presente o no, asociado a la alternativa condicional); y el regreso hasta el casillero de inicio, asociado a la repetición condicional. En la elaboración de la solución completa fue fundamental definir procedimientos para separar cada uno de estos problemas para construir una solución legible.



Solución sin procedimientos para *Contando astros*. A medida que la complejidad de los problemas aumenta es más importante cuidar la legibilidad de la solución.

También nos interesa señalar en cada desafío el análisis del escenario para identificar características que permanecen constantes de aquellas que varían y cómo impacta cada una en la estrategia de solución y la herramienta de programación utilizada para resolverlo. Por ejemplo, podemos utilizar repetición condicional para recorridos de longitud variable o para tareas que consisten en repetir una subtarea una cantidad variable de veces (como resolver la columna con los astros), que no podemos saber cuántas son a *priori*, pero sí identificar cuándo debemos terminar de repetir. Para subproblemas variables que implican una decisión puntual (como contar un planeta o una estrella según el contenido de un casillero en Contando astros o avanzar hacia abajo o hacia la derecha según dónde está el próximo casillero en ***Barrilete cósmico***), utilizamos la alternativa condicional.

Actividad 4
Más repeticiones en nuestro juego

En esta actividad, las y los estudiantes continúan trabajando autónomamente para agregarle al juego puntaje y un objetivo a capturar. También pueden profundizar su trabajo para personalizar el juego y agregarle nuevos comportamientos. Estos desafíos requerirán que recuperen variadas herramientas de programación y las adapten a este contexto particular.

Objetivo >

Utilizar la repetición condicional combinada con otras herramientas de programación para resolver problemas en entornos abiertos abiertos de enseñanza de programación.

Inicio >

El **propósito de este momento** es recuperar el proyecto de la Actividad 2 para continuar trabajando sobre él.

Orientaciones

Recordamos lo hecho en la **Actividad 2**, podemos mostrar nuevamente el videojuego terminado. Brindamos una instancia para que los grupos recuperen su proyecto. Si no está disponible, podemos distribuir el proyecto Andrómeda_meteorito1.sb3 que contiene una solución mínima a los desafíos de la **Actividad 2** y sobre la que podrán seguir trabajando.

Desarrollo >

El **propósito de este momento** es analizar desafíos para identificar, en la construcción de la estrategia de solución, dónde interviene la repetición condicional y cómo se combina con otras herramientas, con mayor autonomía que en las actividades anteriores.

Orientaciones

Invitamos a los grupos a retomar el programa realizado en la **Actividad 2** y añadir nuevas características.

Desafío "Agregar un objetivo". Agregar un objetivo (que se desplace como la estrella en el ejemplo) que el personaje debe alcanzar para sumar un punto. Al alcanzarse 10 puntos, termina el juego.

Antes de comenzar a programar, y como parte del análisis de problema, podemos volver a mirar el juego terminado para analizar el comportamiento que hay que replicar. En cuanto al movimiento, podemos observar que aparece en una posición aleatoria cerca del borde derecho, se mueve hacia el borde izquierdo y cuando lo alcanza, vuelve a empezar cerca del borde derecho. En cuanto al puntaje, cuando el objetivo es alcanzado por el personaje, se suma un punto y el objetivo se reubica en su posición inicial; cuando se alcanzan 10 puntos, el juego termina. Como un comportamiento opcional podemos proponer que la estrella demore una cantidad aleatoria de tiempo en volver a aparecer.

Habilitamos el trabajo autónomo y recorremos los grupos para resolver dificultades. Hacemos especial énfasis en que planteen una estrategia antes de avanzar para que analicen el desafío y puedan identificar subproblemas conocidos (como contar puntaje) y otros que expresen una repetición condicional (por ejemplo, que el objetivo repite su movimiento hasta que se alcancen 10 puntos).

```
Al comenzar:
  Esperar y aparecer cerca del borde derecho
  Poner puntaje en cero
  Repetir hasta que Puntos = 10:
  |  Mover el objetivo
  Decir "Felicitaciones, ganaste"
  Detener todo.

Mover el objetivo:
  Mover a la izquierda
  Reubicar si toca el borde
  Reubicar y sumar puntos si toca el personaje

Reubicar si toca el borde:
  Si ¿Tocando borde izquierdo?:
  |  Esperar y aparecer cerca del borde derecho

Reubicar y sumar puntos si toca el objetivo:
  Si ¿Tocando el objetivo?:
  |  Sumar 1 punto
  |  Esperar y aparecer cerca del borde derecho

Esperar y aparecer cerca del borde derecho:
  Esconder
  Esperar 15 segundos
  Ir cerca del borde derecho
  Mostrar

Ir cerca del borde derecho:
  Ir a X: entre 80 y 200, Y: entre -130 y 130
```

Una estrategia para resolver el desafío con fuerte énfasis en la división en subproblemas.

Resuelto el desafío, proponemos dedicar tiempo para que los grupos profundicen el trabajo en su videojuego. Presentamos algunos desafíos de programación adicionales para ello.

Desafío "Agregar vidas". El personaje empieza con una cantidad de vidas y pierde una cada vez que choca con un enemigo.

Para resolver este desafío alentamos a revisar la estrategia para la programación de los enemigos (meteoritos) y pensar posibles modificaciones.

```
Al comenzar:
  Fijar vidas en 3
  Repetir hasta que vidas < 0:
    Repetir hasta que ¿Tocando al personaje?:
      Mover un paso a la izquierda
      Si ¿Tocando borde izquierdo?:
        Reubicar cerca del borde derecho
    Restar 1 a vidas
  Decir "Perdiste"
  Detener todo
```

```
Al comenzar:
  Fijar vidas en 3.
  Repetir por siempre:
    Avanzar a la izquierda y
      perder una vida si toca el personaje
    Reubicar cerca del borde derecho

Avanzar a la izquierda y perder una vida si toca el personaje:
  Repetir hasta que ¿Tocando el borde izquierdo?:
    Mover un paso a la izquierda
    Si ¿Tocando al personaje?:
      Si vidas = 0:
        Decir Perdiste.
        Detener todo.
      si no:
        Restar 1 a vidas.
```

Dos estrategias para resolver el problema. En color, se destaca la implementación del contador de vidas y de la condición de finalización del juego.

Desafío "Agregar más enemigos". Algunos pueden moverse más rápido o restar más vidas.

Si no queremos modificar el comportamiento de los enemigos (es decir, que sigan apareciendo y avanzando hacia la izquierda), esta consigna no representa un desafío desde el punto de vista de programación, ya que se puede resolver duplicando el objeto del enemigo (haciendo clic derecho y eligiendo "Duplicar"). Una vez que estén duplicados los objetos,

puede modificarse en alguno de ellos la cantidad de vidas que restan o la cantidad de pasos que avanzan en cada repetición.

Clic derecho sobre el objeto para duplicarlo.

Desafío "Personalizar el aspecto". Trabajar en el aspecto del juego, si bien no representa un desafío de programación, es una oportunidad valiosa para que las y los estudiantes se apropien del proyecto y, por lo tanto, puedan identificarse como creadoras y creadores de tecnología. En este sentido, pueden profundizar el trabajo en la parte visual, agregar sonidos, nuevos disfraces para los objetos o mensajes de interacción con el usuario.

También es valioso proponer nuevos comportamientos y ensayar estrategias para conseguirlos. Es una experiencia muy rica reconocer que con los conocimientos que poseen hasta el momento pueden personalizar aún más el comportamiento de sus personajes y sus videojuegos, materializando ideas propias.

Cierre >

El **propósito de este momento** es recuperar los desafíos abordados en las actividades de la secuencia para identificar los distintos problemas o situaciones que resolvieron con la repetición condicional.

Orientaciones

Invitamos a los grupos a que cuenten cómo quedaron sus programas y, en particular, cómo aparece la repetición condicional y cómo se combina con otras herramientas. Podemos ir tomando nota para identificar algunos esquemas generales. La idea es motivar una discusión similar a la del cierre de la actividad anterior, pero sobre los desafíos en el entorno abierto. Además, alentamos a identificar analogías entre los problemas planteados por los desafíos de esta actividad, los desafíos de Pilas Bloques resueltos en la **Actividad 3** y los comportamientos del videojuego que identificaron al inicio de la **Actividad 2**.

Por último, es una buena oportunidad para mirar todo el recorrido de aprendizaje durante las secuencias y repasar entre todas y todos algunas ideas centrales (como por ejemplo, las herramientas de programación aprendidas, pero también la noción de legibilidad, la idea de estrategia y división en subproblemas, la definición y denominación de procedimientos, etc.), para identificar cómo se fueron repitiendo en los sucesivos problemas para proveer una mirada más integral de cada una.

¿Cuáles dirían que fueron las ideas centrales de programación de este recorrido? ¿Cuáles son las herramientas de programación que aprendieron?

Piensen en las diferentes oportunidades en las que las hayan usado (sobre todo cuando las presentamos y cuando las usamos en esta secuencia): ¿para qué problema la necesitaron en cada caso? ¿Se volvieron a encontrar con ese problema después? ¿Dirían que son parecidos estos problemas?

Por ejemplo, podemos recordar la división en subproblemas y la definición de procedimientos que apareció por primera vez en la secuencia didáctica de esa colección "Definimos nuestros bloques" y cómo esa idea se fue completando hasta los desafíos de esta secuencia en la que la división en subproblemas resultó fundamental para generar programas legibles o resolver desafíos cada vez más complejos. Una reflexión análoga cabe para la repetición simple.

Para la alternativa condicional podemos recordar que la presentamos en la secuencia didáctica "¿Cómo se resuelven problemas cambiantes?", con el desafío ***La pelota indecisa***. El escenario cambiaba entre las distintas ejecuciones y, por lo tanto, era necesario obtener información con un sensor para decidir qué acción realizar. Esto volvió a suceder a lo largo de numerosos desafíos y problemas, por ejemplo, para decidir hacia donde avanzar en un recorrido o para detectar si dos objetos estaban colisionando en Scratch.

¿Recuerdan desafíos en los que fue necesario combinar más de una herramienta de programación? ¿Les parece que se pueden resolver problemas interesantes solo con una herramienta? ¿Cómo hicieron para darse cuenta de que tenían que combinar herramientas?

Por ejemplo, los desafíos de las secuencias didácticas de esta colección "Resolvemos recorridos cambiantes" y "Programamos estrategias para problemas cambiantes" requieren combinar alternativa condicional con repetición y procedimientos, ya que distintos aspectos de un mismo

desafío requieren de cada una de ellas: por ejemplo, el desafío ***Festín Astronómico*** requiere evaluar el contenido de cada celda para actuar en consecuencia (y para eso utilizamos la alternativa condicional), requiere repetir un mismo subproblema una cantidad determinada de veces (cada columna de astros, y para eso utilizamos la repetición simple) y requiere utilizar procedimientos para separar los distintos subproblemas dado que se trata de un escenario complejo.

Es valioso reconocer que si entendemos más profundamente cómo funciona y para qué tipo de problemas se puede usar una herramienta, podremos adaptarla y combinarla con otras herramientas para resolver una variedad mayor de problemas.

Material de referencia para docentes

Feria de juegos

Podemos proponer un evento en el que los grupos compartan sus juegos con otros grupos, e incluso, con otras compañeras y otros compañeros de la escuela. Esta experiencia, además de servir para poner de manifiesto la conclusión de un largo recorrido, es una buena oportunidad para motivar la indagación y explicación entre pares. Por ejemplo, analizar el comportamiento del juego de otro grupo es una oportunidad para hipotetizar sobre cómo puede estar programado y responder a esa hipótesis es una oportunidad para analizar el trabajo propio y elaborar explicaciones. Si asisten a la presentación compañeras y compañeros que aún no saben programar, es una oportunidad para motivar este aprendizaje y valorizar estas habilidades entre pares.

ISBN 978-987-48926-3-8

<Program.AR/> Fundación SADOSKY CAF BANCO DE DESARROLLO DE AMÉRICA LATINA Y EL CARIBE