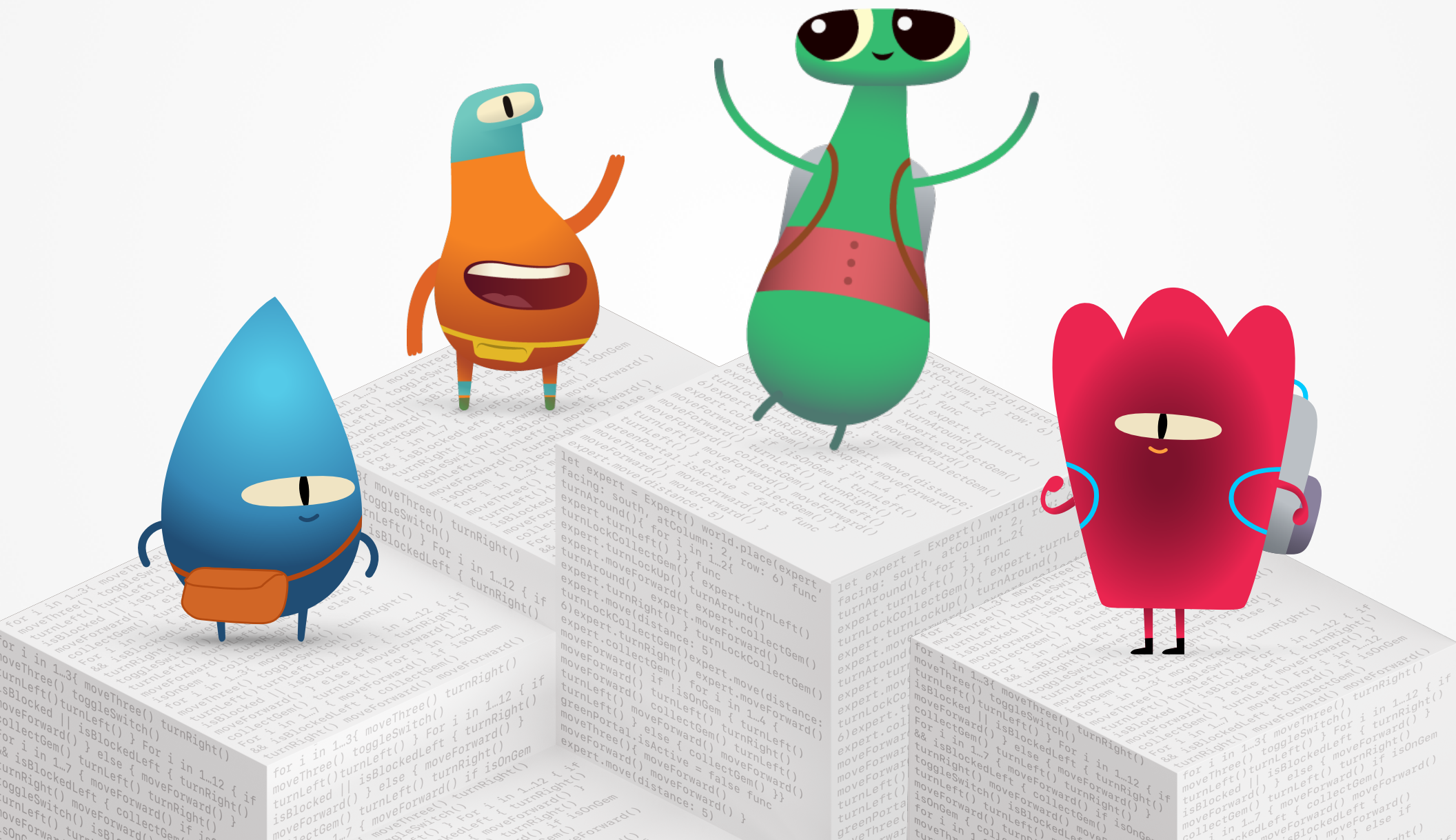




# Programación para todos

Swift Coding Club



# ¡Te damos la bienvenida al Swift Coding Club!

Aprender a programar te enseña a resolver problemas, a trabajar en equipo de formas creativas y a dar vida a tus ideas.

Los Swift Coding Clubs son una forma divertida de aprender a programar y diseñar apps. Con las actividades basadas en Swift, el lenguaje de programación de Apple, trabajas en equipo mientras aprendes a programar, diseñas prototipos de apps y reflexionas sobre cómo cambiar el mundo que te rodea escribiendo código.

No hace falta que seas docente o especialista en programación para tener un Swift Coding Club. Puedes trabajar los materiales a tu ritmo, lo que te permite aprender junto con el resto de integrantes del club. Y podéis celebrar vuestras ideas y diseños organizando una actividad social para presentar las apps que hayáis creado.

Esta guía se divide en tres secciones:



## Empieza

Todo lo que necesitas para montar un Swift Coding Club



## Aprende y aplica

Módulos y actividades para las sesiones del club



## Celebra

Recursos útiles para organizar un evento público

## Recursos de programación

Los Swift Coding Clubs giran en torno a una gran variedad de recursos pensados para aprender programación. Apple acompaña a los programadores desde el aprendizaje de los fundamentos hasta la creación de apps reales.



### Programación para todos | A partir de 10 años

Usa código Swift para aprender los fundamentos de la programación con Swift Playgrounds en un iPad o un Mac. [Más información >](#)



### Desarrolla en Swift | A partir de 14 años

Aprende a desarrollar apps con Xcode en el Mac. [Más información >](#)



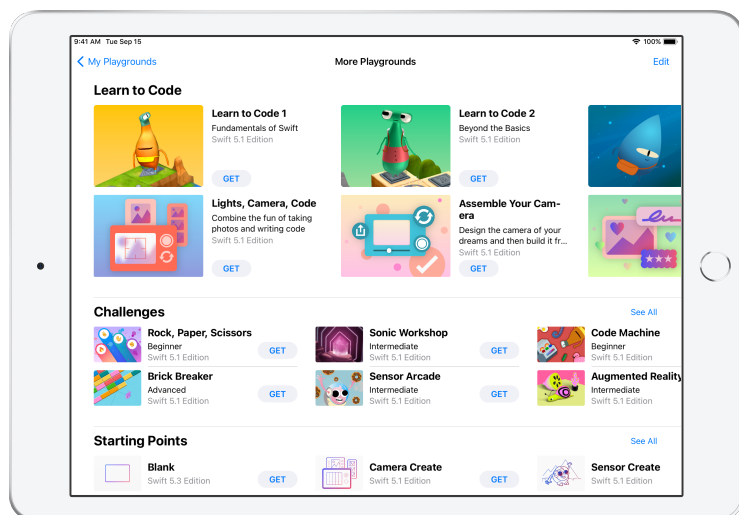
# Empieza

## 1. Echa un vistazo a los recursos de «Programación para todos»

«Programación para todos» inicia a los estudiantes en el mundo de la programación a través de rompecabezas interactivos, personajes divertidos y actividades interesantes. Antes de empezar a dar forma a la dinámica del club, te vendrá bien consultar los siguientes recursos de «Programación para todos».

Swift Playgrounds es una app gratuita para aprender código Swift de forma divertida e interactiva. Incluye una biblioteca de lecciones y retos adicionales creados por desarrolladores y editores de primer nivel.

Las guías de «Programación para todos» incluyen actividades para aprender conceptos básicos de programación, asociarlos a la vida cotidiana y aplicarlos resolviendo rompecabezas en Swift Playgrounds.



[Descarga y explora Swift Playgrounds >](#)



[Descarga la colección «Programación para todos» >](#)



## 2. Recopila la tecnología que necesitaréis

Asegúrate de tener todo esto antes de la primera reunión:

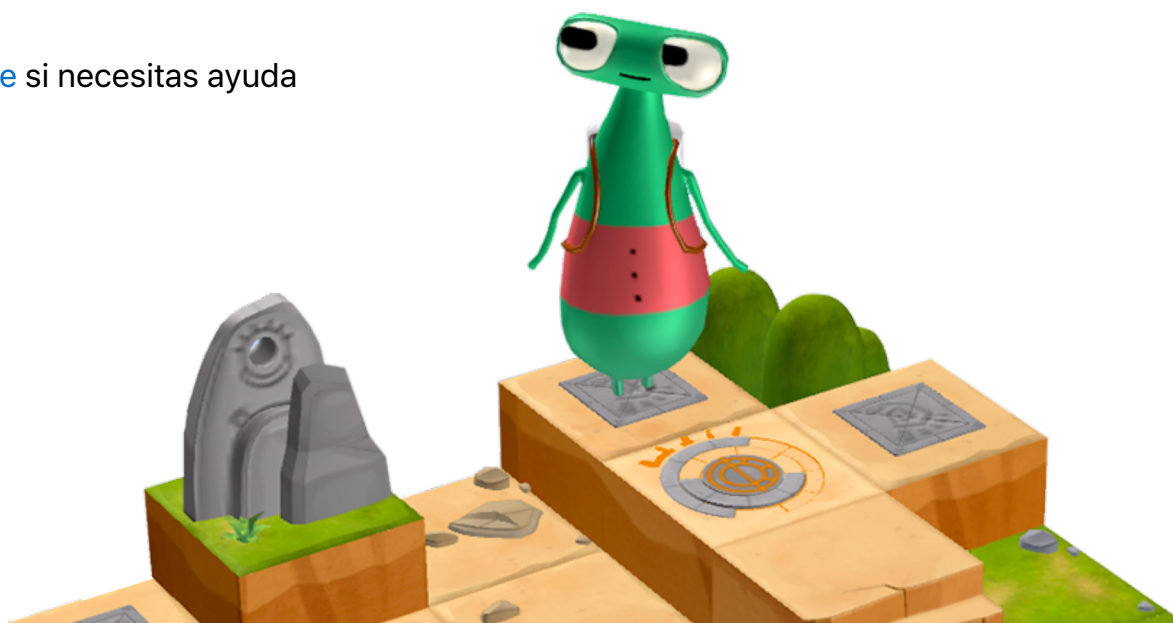
- **iPad o Mac.** Swift Playgrounds requiere dispositivos iPad con iPadOS 13 o una versión posterior, o bien dispositivos Mac con macOS 10.15.3 o una versión posterior. Lo mejor es que cada persona tenga su propio dispositivo, pero también podéis compartir y programar juntos.
- **App Swift Playgrounds.**  
[Descarga Swift Playgrounds para iPad >](#)  
[Descarga Swift Playgrounds para Mac >](#)
- **Guías de «Programación para todos».**  
[Descarga «Programación para todos — Rompecabezas» >](#)  
[Descarga «Programación para todos — Aventuras» >](#)  
(opcional)

Visita la web [Soporte técnico de Apple](#) si necesitas ayuda con los productos Apple.

## 3. Traza un plan

Ten en cuenta lo siguiente:

- ¿Quiénes forman parte del club? ¿Qué cosas les interesan? ¿Tienen experiencia en programación o son principiantes?
- ¿Con qué frecuencia se va a reunir el club? Si estás planificando un campamento de verano, ¿cuántas horas vais a dedicar a las actividades de programación?
- ¿Qué tecnología está a disposición del club?
- ¿Cuáles son los objetivos del club?





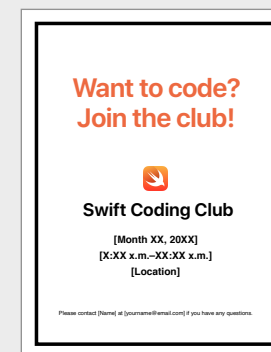
## 4. Que corra la voz

Da a conocer tu Swift Coding Club. Aquí tienes algunas ideas y recursos para atraer a nuevos miembros a tu club:

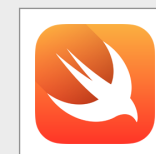
- **Anuncia el club.** Usa el correo electrónico, redes sociales, internet, folletos y el boca a boca para que la gente conozca tu club.
- **Organiza una reunión informativa.** Pregunta a las personas que estén interesadas en formar parte del club qué cosas les interesan y qué tipos de proyectos les gustaría crear. Propón ideas para organizar eventos públicos y cómo se pueden involucrar los miembros del club. También puedes subir a internet un pequeño vídeo sobre el club.

Los siguientes materiales pueden ayudarte a promocionar y personalizar tu Swift Coding Club:

- **Carteles.** [Descarga esta plantilla gratuita](#) y personalízala para crear tu propio cartel. Imprímelo y exponlo, o haz un cartel digital para subirlo a internet. No olvides indicar cuándo y dónde se reúne el club, y cómo apuntarse.
- **Pegatinas y camisetas.** Usa estas [pegatinas del Swift Coding Club](#) para promocionar tu club. Las camisetas son muy útiles para identificar a quienes participan en los eventos de presentación de las apps. Descarga la [plantilla para camisetas del Swift Coding Club](#) y haz camisetas para los miembros.



Cartel del Swift Coding Club



Pegatina del Swift Coding Club



Camiseta del Swift Coding Club

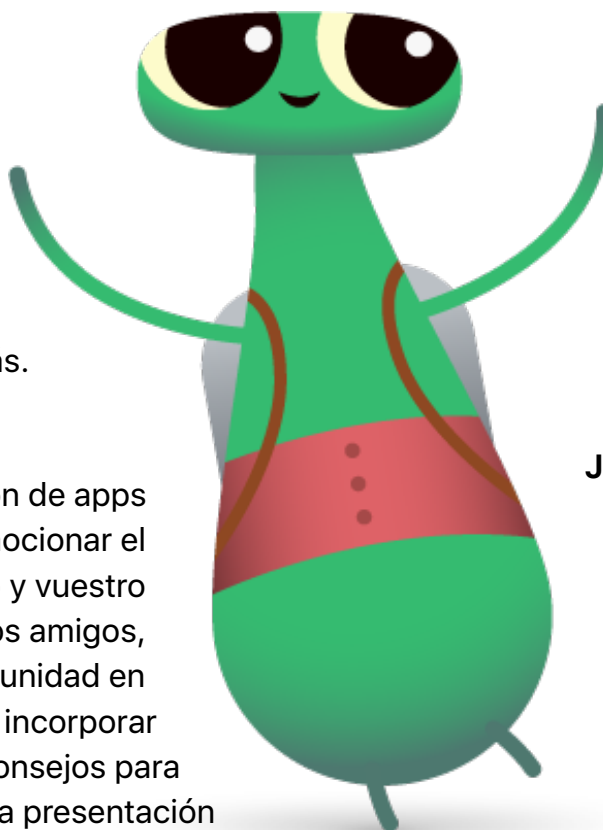
## Consejos para responsables del club



**Monta un equipo de dirección.** Si tienes un equipo de personas que te ayuden a dirigir el club, todo será mucho más fácil y divertido. ¿Quién tiene dotes de mando? Estudia la posibilidad de designar responsables de coordinar los eventos, la programación, el diseño de apps, etc.

**Aprende juntos.** Los responsables de un club no tienen por qué saberlo todo. Ayuda a los miembros a investigar por su cuenta y mejorar sus habilidades de resolución de problemas, y anímalos a ayudar a los demás.

**Presume.** Un evento o presentación de apps es una gran oportunidad para promocionar el club, vuestras ideas de diseño y vuestro talento para programar entre los amigos, familiares, profesores y la comunidad en general. Incluso puede servir para incorporar a nuevos miembros. Encontrarás consejos para organizar un evento público o una presentación de apps en la [página 12](#).



**Comparte ideas.** A algunos miembros les interesará hacer juegos. Otros preferirán crear apps para ayudar a la gente, aprender Swift o controlar robots. Busca formas para que los miembros colaboren en proyectos que les interesen.

**Júntalos.** A veces los miembros más aventajados pueden dejar atrás a los rezagados. Procura emparejar a los primeros con los segundos para que programen juntos. ¡Enseñar a alguien es una forma estupenda de aprender!

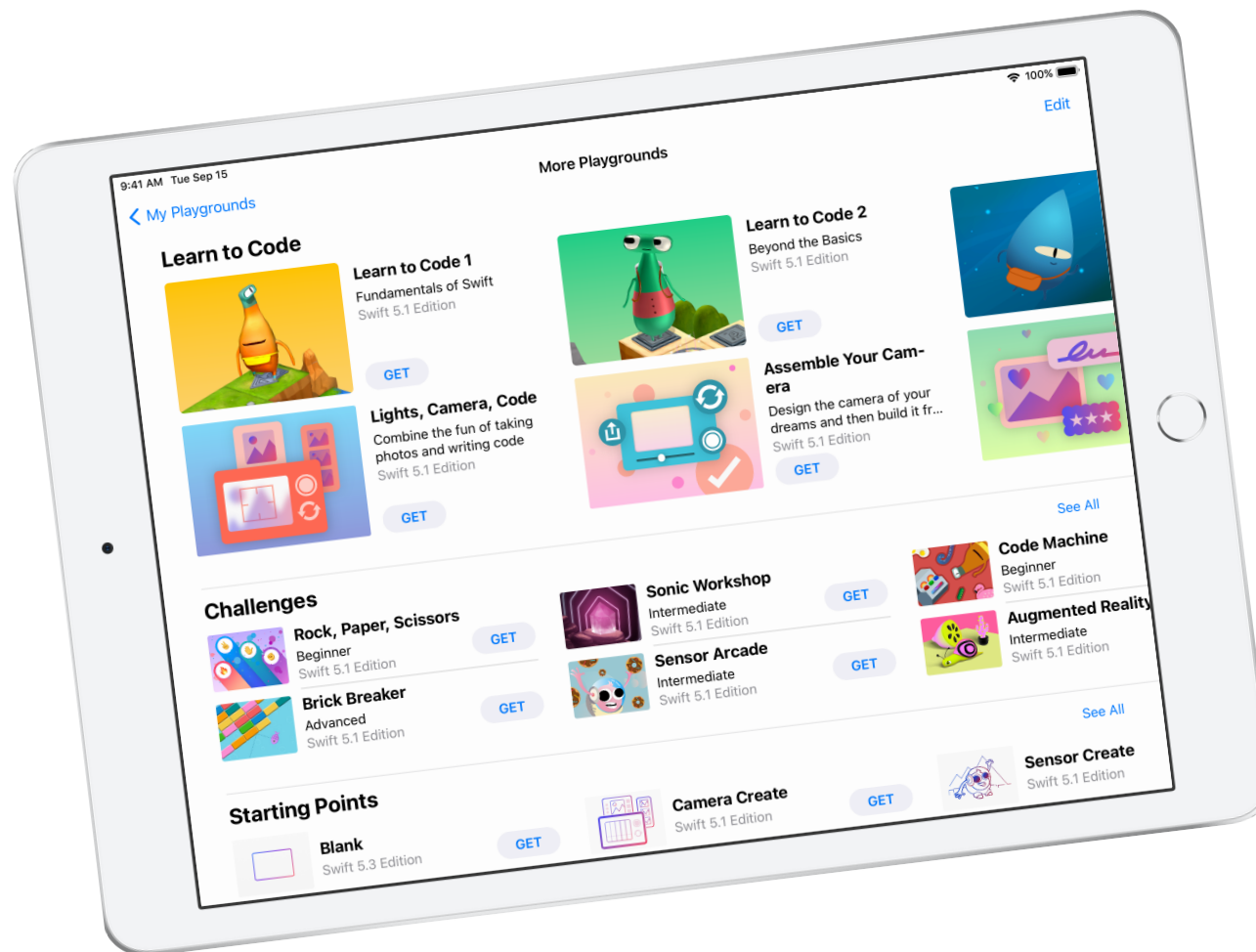


# Aprende y aplica



## 1. Explora Swift Playgrounds

Los materiales del club están pensados para Swift Playgrounds, que incluye una biblioteca de lecciones y retos adicionales creados por desarrolladores y editores de primer nivel. Empieza por familiarizarte con el contenido de Swift Playgrounds y las prestaciones de la app.



# Prestaciones de Swift Playgrounds



## Biblioteca de fragmentos

Para escribir lo mínimo posible, toca en la barra de herramientas para acceder a la biblioteca de fragmentos y arrastra los fragmentos de código más utilizados.

## Herramientas

Usa este menú para restablecer la página, hacer una foto, crear un PDF o grabar un vídeo.

## Menú de páginas

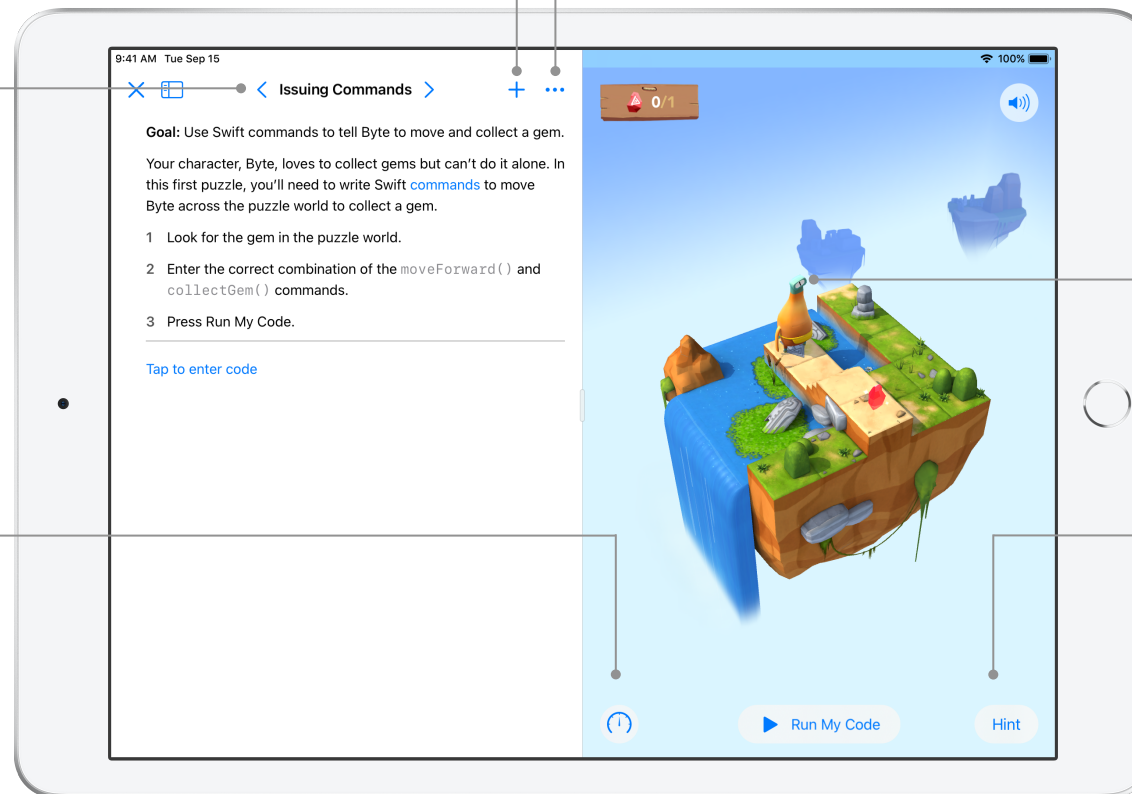
Toca el encabezado de la página para ver todas las páginas del área de juegos. Toca una página o usa las flechas para pasar de página.

## Controla la velocidad

Aumenta o reduce la velocidad a la que se ejecuta el código.

## Resalta el código mientras se ejecuta

Selecciona «Avanzar por mi código» para ir resaltando las líneas de código conforme se ejecutan y comprender mejor lo que hace.



## Elige un personaje

Personaliza la experiencia tocando el personaje para elegir uno diferente.

## Pista

Esta prestación hace sugerencias muy interesantes. Aunque al final muestra la solución al rompecabezas, no basta con cortarla y pegarla: para avanzar es necesario completar los pasos y escribir el código.

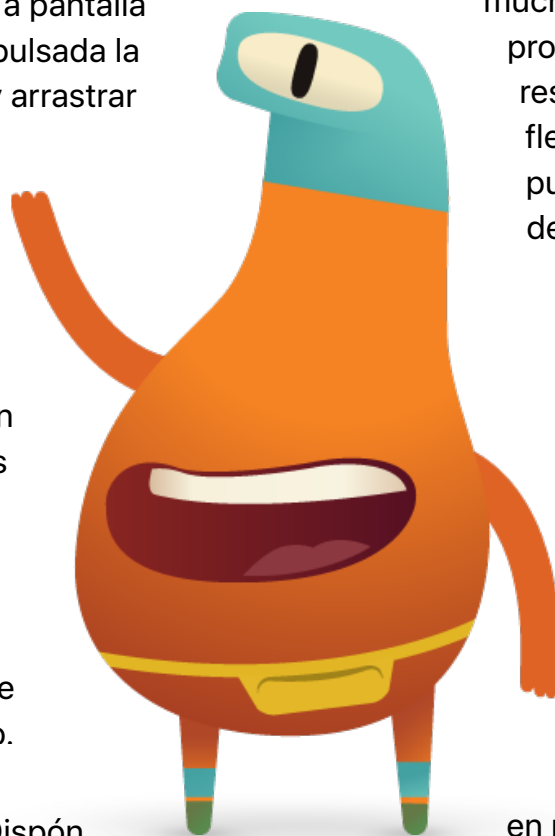
# Consejos para aprender con Swift Playgrounds



**Empieza explorando los rompecabezas.** Anima a los miembros del club a acercar la vista y girar el mundo de Byte en la vista en directo para que puedan ver bien lo que tienen que hacer. Si quieren verlo a pantalla completa, pueden mantener pulsada la división entre las dos ventanas y arrastrar hacia la izquierda.

**Divide los rompecabezas en partes más pequeñas.** Los rompecabezas se van poniendo cada vez más difíciles. Los miembros del club pueden dividir un rompecabezas en partes más pequeñas para que les resulte más fácil pensar en todos los pasos que tienen que dar para resolverlo. Pueden usar Pages o Notas para planificar y escribir los pasos antes de introducir el código.

**Monta un puesto de ayuda.** Dispón un espacio donde los expertos del club puedan ayudar a sus compañeros.



**Busca varias soluciones.** Cada rompecabezas tiene muchas soluciones. Si los miembros terminan pronto, anímalos a buscar diferentes formas de resolver los rompecabezas. Pensar de forma flexible y comparar diferentes soluciones puede ayudarlos a mejorar sus habilidades de pensamiento crítico.

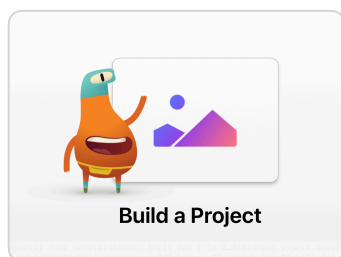
**Programación por parejas.** Invita a los miembros del club a trabajar juntos en un iPad o un Mac. Pueden proponer ideas para resolver los rompecabezas y luego turnarse para escribir el código.

**Usa prestaciones de accesibilidad.** Swift Playgrounds funciona de maravilla con las prestaciones de accesibilidad integradas en macOS y iPadOS para que todo el mundo pueda aprender a programar. Por ejemplo, los programadores pueden invertir los colores, activar la escala de grises o ajustar la visibilidad con el zoom.

## 2. Elige los módulos

Los materiales del club se organizan en módulos que combinan actividades de programación y diseño. Cada módulo consta de 12 sesiones de una hora y aborda un tema y un nivel de experiencia concretos. En las sesiones «Aprende y prueba», los miembros del club exploran conceptos clave y los aplican programando rompecabezas y retos en Swift Playgrounds. En las sesiones «Aplica y conecta», analizan cómo usar el código para explorar ideas y crear nuevos productos. Ponen en práctica sus conocimientos de diseño y programación para crear o diseñar un proyecto de Swift Playgrounds para un público específico.

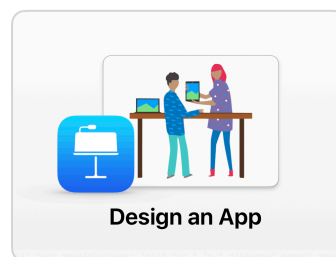
Encontrarás guías para cada módulo en la segunda parte de este documento, aunque también puedes usar los siguientes enlaces si quieres echar un vistazo ahora.



### Crea un proyecto

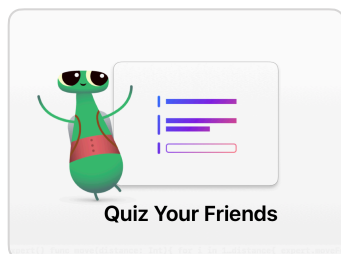
Los miembros del club dominan los aspectos básicos de la programación con Swift Playgrounds en «Aprende a programar 1» y «Aprende a programar 2». Ponen en práctica sus nuevas habilidades para diseñar y crear un proyecto de área de juegos que responda a eventos táctiles.

[Ver el módulo >](#)



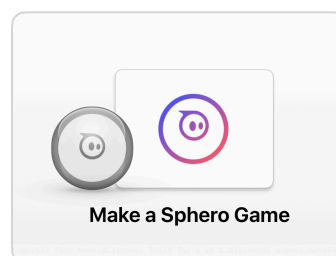
### Diseña una app

Los miembros del club diseñan juntos una app para ayudar a resolver un problema de su entorno. Durante el proceso aprenden a hacer lluvias de ideas, planificar, hacer un prototipo y evaluar una app original. [Ver el módulo >](#)



### Pregunta a tus amigos

Los miembros del club siguen perfeccionando lo que aprendieron en «Crea un proyecto» completando rompecabezas de Swift Playgrounds más complicados en «Aprende a programar 1» y «Aprende a programar 2». Crean un proyecto de área de juegos que solicita y responde a la información del usuario. [Ver el módulo >](#)



### Haz un juego de Sphero

Los miembros del club programan un robot Sphero para recrear el típico juego de máquina recreativa. Analizan juntos el código que hay detrás del juego y lo editan para adaptarlo. Usan sus conocimientos para diseñar su propio juego con uno o más robots Sphero. [Ver el módulo >](#)





### 3. Ve más allá

También puedes añadir sesiones sobre cosas que interesen a los miembros del club. Puedes ir más allá con las actividades de diseño de apps y programación: prueba con un dispositivo conectado, crea una carrera de obstáculos para un dron o prepara una misión de rescate para un robot.

Para fomentar la reflexión sobre las cuestiones de diseño, puedes invitar también a ponentes especiales u organizar excursiones para ayudar a los miembros del club a entender mejor al público y las necesidades de diseño de un proyecto.



# Celebra



## Evento público o presentación de apps

Organiza un evento público o una presentación de apps para implicar a todo el mundo. Así podrás transmitir las posibilidades que ofrece el código a la hora de resolver problemas actuales. Estos eventos también son una ocasión ideal para demostrar el talento de los miembros del club.

**1. Planifica el gran evento:** Elige una fecha e invita a estudiantes, docentes, madres y padres, y demás miembros de la comunidad.

Asigna tiempo a cada equipo para que presente su proyecto y organiza una breve ronda de ruegos y preguntas. Si tienes un grupo numeroso, puedes dividir el club en dos fases, y así también podrán ver las presentaciones de sus compañeros.

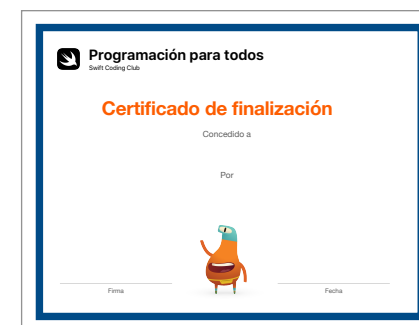
Si quieres, puedes cerrar el evento con un divertido pase de diapositivas de fotos hechas durante las sesiones del club.



**2. Crea premios:** La competencia sana puede ser una gran fuente de motivación. Incentiva a los miembros del club con unos premios que reconozcan sus logros en la programación y el diseño de su app:

- Mejor ingeniería
- Mejor innovación
- Mejor diseño
- Mejor presentación

También podrías animar al público a participar con el Premio del Público.



Puedes descargar y modificar este [certificado](#) para los distintos premios.

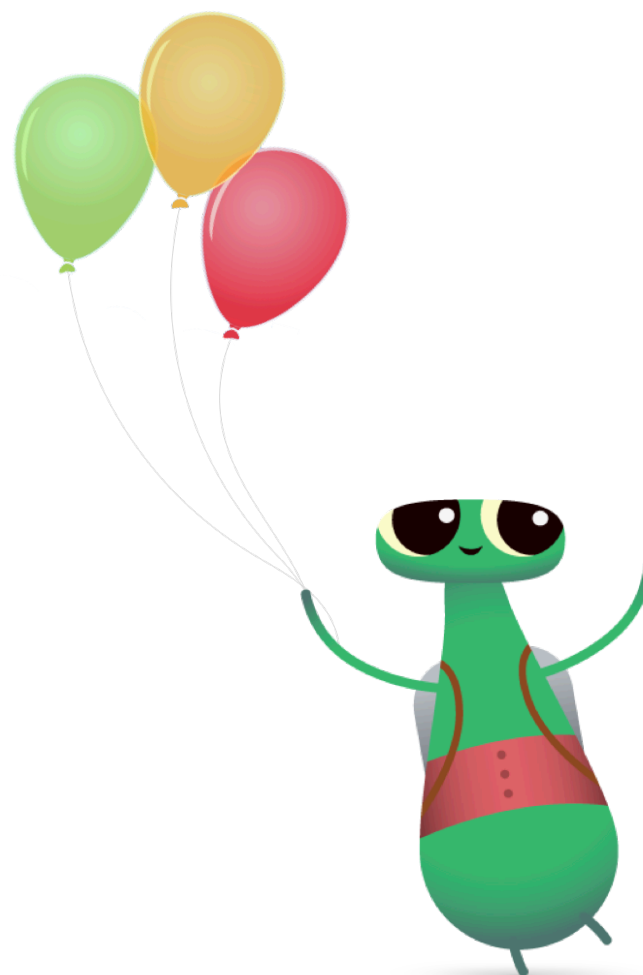
**3. Selecciona al jurado y los mentores.** Los miembros del jurado y los mentores pueden ser docentes o empleados del centro, estudiantes con experiencia en programación, especialistas del sector del desarrollo o el diseño, integrantes de la junta escolar, personalidades locales o personas a las que pueda beneficiar la nueva idea.

No es necesario que el jurado espere hasta la presentación para reunirse con el club. Si te parece, puedes traerlos como invitados especiales para que compartan sus conocimientos cuando los estudiantes estén trabajando en la lluvia de ideas o la planificación del diseño de su proyecto.

**4. Comparte e inspira.** Puedes grabar las presentaciones. Compártelas con la comunidad local y crea un vídeo con los momentos destacados para inspirar a futuros miembros del club.

Consulta la «Guía de presentación de apps» para obtener más inspiración y consejos para los eventos de presentación.

[Descarga la «Guía de presentación de apps» >](#)





# Programación para todos

Swift Coding Club

## Certificado de finalización

Concedido a

Por



---

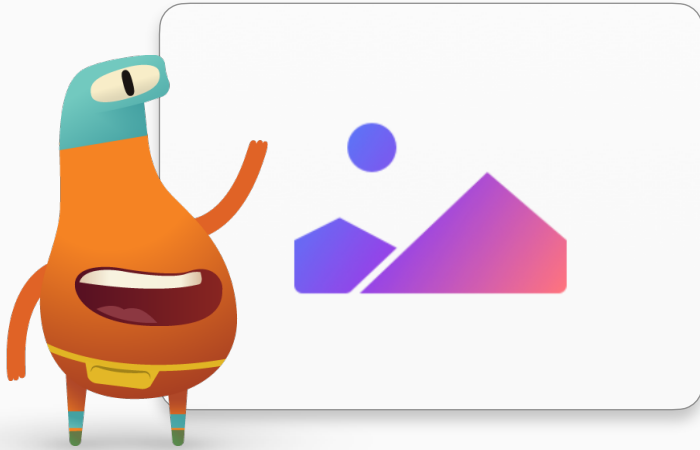
Firma

---

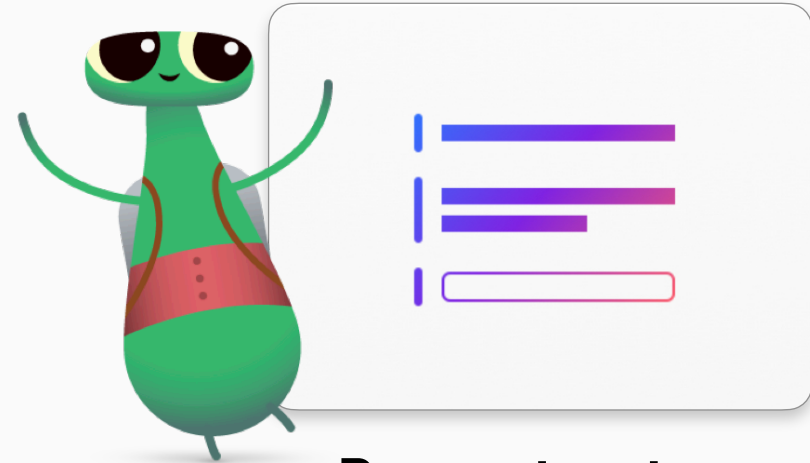
Fecha



# Módulos del Swift Coding Club



**Crea un proyecto**



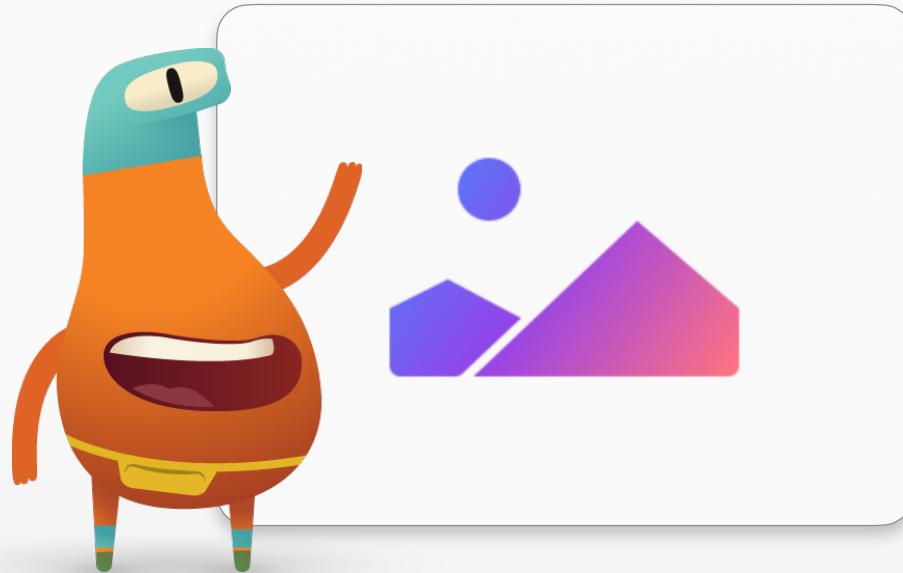
**Pregunta a tus amigos**



**Diseña una app**

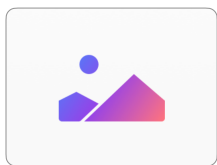


**Haz un juego de Sphero**



# Crea un proyecto

```
func show(from actor: Actor) {guard state == .inactive else { return }state = .animatingToPicker// Use the idle for the c
commandSpeed = WorldConfiguration.Actor.idleSpeedoriginalWorldActor = actororiginalActorTransform =
sceneNode.transformactor.reset()actor.sceneNode.runAction(.playSoundEffect(tapSource)) // Reset the existing `pickerActors`.
d).for pickerActor in pickerActors {pickerActor.reset()pickerActor.isInCharacterPicker = true} let result = actor.perform
result.completionHandler = { _ inguard self.state == .animatingToPicker else TriggerDepthOfField(intro: true) { [unowned
loadAndDisplayCharacters()}}}func triggerDepthOfField(intro: Bool, completion: CompletionBlock? = nil) {let root = scene?.r
node = root?.childNode(withName: "camera", recursively: true),let camera = cameraNode.camera else { return } SCNTransaction
saction.animationDuration = CharacterPickerController.fadeDurationcamera.focusDistance = intro ? focusDistanceMax : 0came
de : fStopDefault SCNTransaction.completionBlock = completionSCNTransaction.commit()private func loadAndDisplayCharacters
return } / Setup overlay view & constraooverlayView.alpha = 0overlayView.autoresizingMask = [.flexibleWidth, .flexibleHeig
ldSubview(overlayView).view.bounds // Add a gesture recognizer to detect when character selection happens.let tapGesi
ction: #selector(selectCharacter(:)) overlayView.gestureRecognizers = [tapGesture // Setup overlay scene
```



# Crea un proyecto




## Resumen del módulo

En estas sesiones, los miembros del club dominarán las nociones básicas de la programación haciendo divertidas actividades de la guía «Programación para todos — Rompecabezas». Practicarán con el código resolviendo rompecabezas de «Aprende a programar 1» y «Aprende a programar 2» de Swift Playgrounds y pondrán en práctica sus nuevas habilidades para diseñar y crear un proyecto de área de juegos que responda a eventos táctiles.

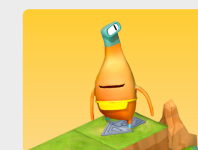
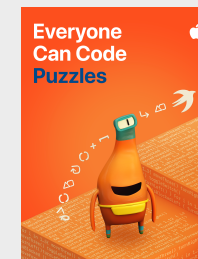
En las sesiones «Aprende y prueba», los miembros del club explorarán conceptos clave y los aplicarán programando rompecabezas y retos en Swift Playgrounds. En las sesiones «Aplica y conecta», aprenderán a usar el código para desarrollar ideas y crear productos nuevos. Al final de las sesiones, plantéate organizar un evento público para que los miembros del club expongan sus proyectos.

En las actividades se incluyen los números de página de la guía «Programación para todos — Rompecabezas». Para obtener más información sobre cada actividad, acceder a recursos adicionales y descubrir cómo apoyar o retar a los miembros del club, consulta la [Guía para profesores de «Programación para todos — Rompecabezas»](#).

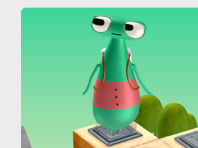
### Resumen de sesiones

-  Aprende y prueba: 6 sesiones
-  Aplica y conecta: 6 sesiones
-  Evento público

## Recursos



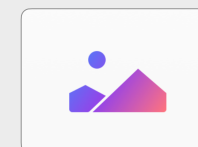
**Aprende a programar 1**



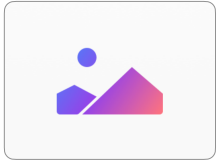
**Aprende a programar 2**



**Espirales**



**Figuras**



# Crea un proyecto

## 1 Comandos

Descubre el concepto básico de comando: una instrucción específica que se da a un ordenador. Aprende a programar usando una secuencia de comandos.

**Aprende:** Echa un vistazo a la introducción a los comandos de «Aprende a programar 1».

El escondite (página 3)

**Prueba:** Completa los rompecabezas del capítulo dedicado a los comandos de «Aprende a programar 1» (páginas 4–10).

### Aprende a programar 1 Comandos



- Introducción
- Dar órdenes con comandos
- Añadir un comando nuevo
- Activar los interruptores

## 2 Funciones

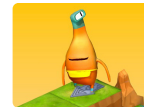
Aprende a crear tus propios comandos creando funciones y llamándolas.

**Aprende:** Echa un vistazo a la introducción a las funciones de «Aprende a programar 1».

Origami (página 15)

**Prueba:** Completa los rompecabezas del capítulo dedicado a las funciones de «Aprende a programar 1» (páginas 16–21).

### Aprende a programar 1 Funciones



- Introducción
- Componer un comportamiento
- Crear una función nueva
- Patrones anidados

## 3 Bucles for

Explora los bucles for y cómo programar de forma más eficaz con las funciones y los bucles.

**Aprende:** Echa un vistazo a la introducción a los bucles for de «Aprende a programar 1».

Creador de patrones (página 26)

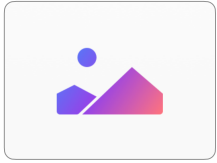
**Prueba:** Completa los rompecabezas del capítulo dedicado a los bucles for de «Aprende a programar 1» (páginas 27–31).

### Aprende a programar 1 Bucles for



- Introducción
- Utilizar bucles
- Bucles por los cuatro costados





# Crea un proyecto

## 4 Variables

Aprende cómo almacenan la información los ordenadores por medio de variables y cómo utilizarlas para programar.

**Aprende:** Echa un vistazo a la introducción a las variables de «Aprende a programar 2». NewsBot (página 36)

**Prueba:** Completa los rompecabezas del capítulo dedicado a las variables de «Aprende a programar 1» y el reto «Espirales» (páginas 37–43).

### Aprende a programar 2 Variables

- Introducción
- Llevar el recuento



### Espirales

- Descripción
- Hipocicloides
- Epicicloides
- Hipotrocoides
- Elipses
- ¡A jugar!



## 5 Código condicional

Descubre la lógica booleana y aprende a escribir código condicional.

**Aprende:** Echa un vistazo a la introducción al código condicional de «Aprende a programar 1».

Alguien dice (página 49)

**Prueba:** Completa los rompecabezas del capítulo dedicado al código condicional de «Aprende a programar 1» (páginas 50–56).

### Aprende a programar 1 Código condicional

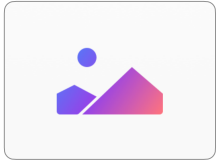
- Introducción
- Comprobación de interruptores
- Utilizar else if
- Código condicional en bucle
- Definir funciones más eficaces



## 6 Diseña para un público

Piensa en los distintos usuarios a los que podría interesar tu app y reflexiona sobre cómo diseñar productos según el público.

**Conecta:** Ponte en la piel de otra persona (página 58).



# Crea un proyecto

## 7 Tipos e inicialización

Aprende a describir tipos y a inicializarlos en tu código.

**Aprende:** Echa un vistazo a los capítulos sobre tipos e inicialización de «Aprende a programar 2».

Cualidades de un buen diseño (página 62)

**Prueba:** Completa los rompecabezas de los capítulos dedicados a los tipos e inicialización de «Aprende a programar 2» (páginas 63–66).

### Aprende a programar 2 Tipos

- Introducción
- Desactivar un portal



### Inicialización

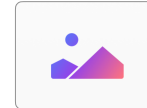
- Introducción
- Inicializar al experto
- Usar instancias de distintos tipos

## 8 Figuras interactivas

Echa un vistazo al punto de inicio «Figuras» de Swift Playgrounds. Desde ahí comenzarás a crear tu proyecto en las siguientes sesiones. Experimenta con las páginas «Gráficos con formas» y «Toques y animaciones», y observa qué sucede con cada código y cómo. En grupo, enumera los elementos gráficos y las funciones disponibles en el punto de inicio «Figuras».

### Figuras

- Gráficos con formas
- Toques y animaciones



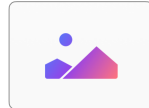
## 9 Crea un proyecto de figuras

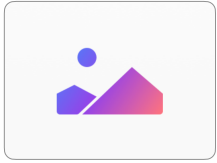
Descubre cómo crear un proyecto para mejorar la coordinación ojo-mano en el punto de inicio «Figuras». Repasa tu lista de elementos gráficos y funciones y añade lo que sea necesario.

**Aplica:** Crea un proyecto para mejorar la coordinación ojo-mano (página 67).

### Figuras

- Toques y animaciones





# Crea un proyecto

## 10 Diseña un proyecto

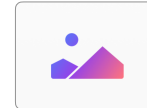
Recopila ideas de otros proyectos que podrías crear con el punto de inicio «Figuras». Ten en cuenta los elementos gráficos y las funciones disponibles y cómo podrían responder a las necesidades de un público concreto. El objetivo es analizar las ideas en grupo y, después, trabajar por parejas para esbozar una idea original que muestre cómo el proyecto se ajusta a tu objetivo y a un público específico.

## 11 Crea el proyecto

Programa por parejas tu idea de proyecto en la página «Toques y animaciones» del punto de inicio «Figuras». Consulta el boceto de la última sesión.

### Figuras

- Toques y animaciones

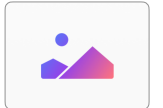


## 12 Evalúa el proyecto

Prueba tu proyecto de área de juegos con tus compañeros. Ensaya cómo vas a explicar el funcionamiento de tu proyecto (y tus decisiones de diseño) en el evento público en el que compartirás tus creaciones.

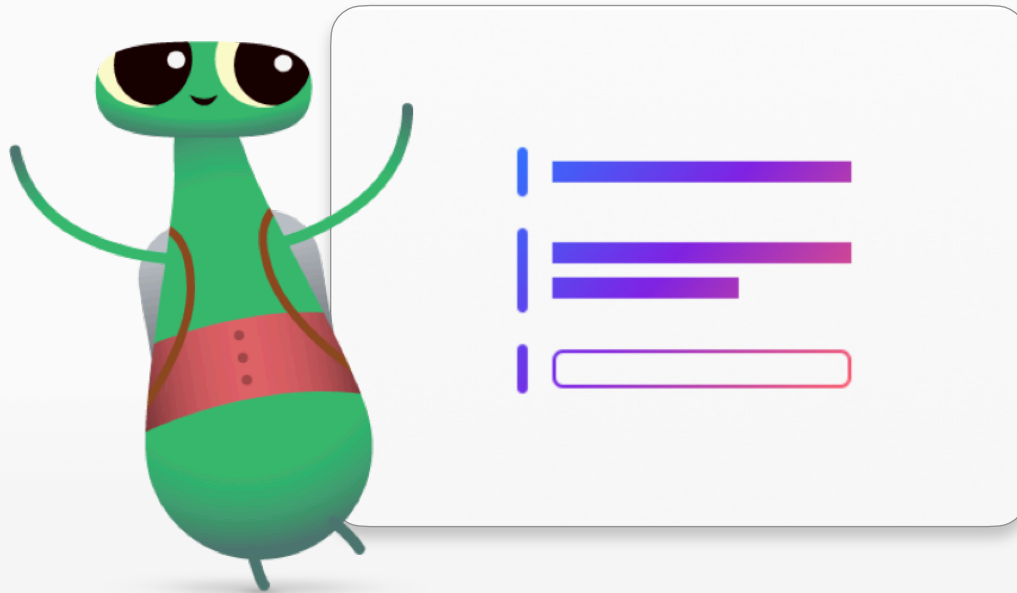
### Figuras

- Toques y animaciones



## Evento público

Celebra los logros del club en un evento público. Así podrás exponer tu proyecto, explicar el proceso de diseño y conocer la opinión de los demás.



# Pregunta a tus amigos

```
func show(from actor: Actor) {guard state == .inactive else { return }state = .animatingToPicker// Use the idle for the c
commandSpeed = WorldConfiguration.Actor.idleSpeedoriginalWorldActor = actororiginalActorTransform =
sceneNode.transformactor.reset()actor.sceneNode.runAction(.playSoundEffect(tapSource)) // Reset the existing `pickerActors`.
d).for pickerActor in pickerActors {pickerActor.reset()pickerActor.isInCharacterPicker = true} let result = actor.perform
result.completionHandler = { _ inguard self.state == .animatingToPicker else TriggerDepthOfField(intro: true) { [unowned
adAndDisplayCharacters()}}}func triggerDepthOfField(intro: Bool, completion: CompletionBlock? = nil) {let root = scene?.r
ode = root?.childNode(withName: "camera", recursively: true),let camera = cameraNode.camera else { return } SCNTransaction
saction.animationDuration = CharacterPickerController.fadeDurationcamera.focusDistance = intro ? focusDistanceMax : 0came
de : fStopDefault SCNTransaction.completionBlock = completionSCNTransaction.commit()private func oadAndDisplayCharacters
return } // Setup overlay view & constraoverlayView.alpha = 0overlayView.autoresizingMask = [.flexibleWidth, .flexibleHeig
ldSubview(overlayView)view.bounds // Add a gesture recognizer to detect when character selection happens.let tapGest
ction: #selector(selectCharacter(:)) overlayView.gestureRecognizers = [tapGesture // Setup overlay sce
```



# Pregunta a tus amigos




## Resumen del módulo

En este módulo, los miembros del club ponen en práctica lo aprendido completando actividades más avanzadas de la guía «Programación para todos — Rompecabezas». Practican su código resolviendo los rompecabezas de «Aprende a programar 1» y «Aprende a programar 2» en Swift Playgrounds y usan sus habilidades avanzadas para desarrollar un proyecto de área de juegos que solicita información al usuario y le responde. Para este módulo es necesario entender los materiales de los capítulos 1-6 de «Rompecabezas», completar el módulo «Crea un proyecto» o tener un nivel equivalente de conocimientos.

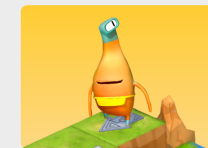
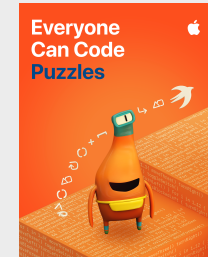
En las sesiones «Aprende y prueba», los miembros del club exploran conceptos clave y los aplican programando rompecabezas y retos en Swift Playgrounds. En las sesiones «Aplica y conecta», aprenderán a usar el código para desarrollar ideas y crear productos nuevos. Al final de las sesiones, plantéate organizar un evento público para que los miembros del club expongan sus proyectos.

En las actividades se incluyen los números de página de la guía «Programación para todos — Rompecabezas». Para obtener más información sobre cada actividad, acceder a recursos adicionales y descubrir cómo apoyar o retar a los miembros del club, consulta la [Guía para profesores de «Programación para todos — Rompecabezas»](#).

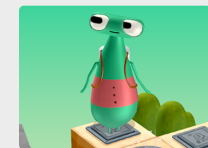
### Resumen de sesiones

-  Aprende y prueba: 4 sesiones
-  Aplica y conecta: 8 sesiones
-  Evento público

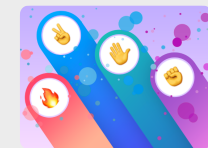
## Recursos



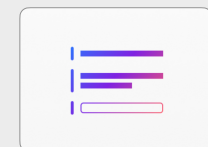
**Aprende a programar 1**



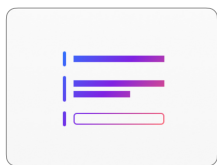
**Aprende a programar 2**



**Piedra, papel o tijera**



**Respuestas**



# Pregunta a tus amigos

## 1 Funciones con parámetros

Aprende a hacer funciones más precisas con parámetros que den más información al ordenador.

**Aprende:** Echa un vistazo a la introducción a los parámetros de «Aprende a programar 2».

La receta del éxito (página 71)

**Prueba:** Completa los rompecabezas del capítulo dedicado a los parámetros de «Aprende a programar 2» (páginas 72–75).

### Aprende a programar 2 Parámetros

- Introducción
- Seguir avanzando



## 2 Diseña un juego

Usa el reto de piedra, papel o tijera de Swift Playgrounds para diseñar una nueva edición mejorada del juego.

**Aplica:** Crea un juego de piedra, papel o tijera (página 76).

### Piedra, papel o tijera

- Crear un juego
- Compartir código
- Añadir acciones
- Modificar propiedades



## 3 Operadores lógicos

Aprende a programar comportamientos concretos en respuesta a ciertas condiciones gracias a los operadores lógicos.

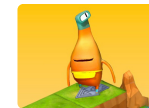
**Aprende:** Echa un vistazo a la introducción a los operadores lógicos de «Aprende a programar 1».

Alguien dice, segunda ronda (página 81)

**Prueba:** Completa los rompecabezas del capítulo dedicado a los operadores lógicos de «Aprende a programar 1» (páginas 82–85).

### Aprende a programar 1 Operadores lógicos

- Introducción
- Utilizar el operador NO
- Comprobar esto Y aquello
- Comprobar esto O aquello





# Pregunta a tus amigos

## 4 Crea un cuestionario

Pon en práctica todo lo que sabes sobre operadores con condiciones, variables, funciones y funciones con parámetros para crear un cuestionario en el punto de inicio «Respuestas» de Swift Playgrounds.

**Aplica:** Crea un cuestionario (página 86).

## 5 Diseña un proyecto de cuestionario

Piensa en una idea para tu proyecto de cuestionario, creado a partir del punto de inicio «Respuestas». Determina la finalidad del cuestionario, analiza diseños para las apps de cuestionario, ten en cuenta el público objetivo y esboza tu idea.

## 6 Bucles while

Descubre los bucles while y cómo usarlos para repetir un bucle de código hasta que la condición sea verdadera.

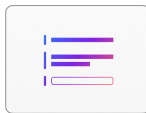
**Aprende:** Echa un vistazo a la introducción a los bucles while de «Aprende a programar 1».

Juegos del recreo (página 90)

**Prueba:** Completa los rompecabezas del capítulo dedicado a los bucles while de «Aprende a programar 1» (páginas 91–94).

## Respuestas

- Texto
- Resumen de API
- Tipos

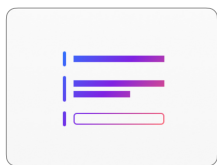


## Aprende a programar 1 Bucles while



- Introducción
- Ejecutar código mientras...
- Crear bucles while más inteligentes
- Anidar bucles





# Pregunta a tus amigos

## 7 Pule el cuestionario

Modifica el cuestionario original para introducir distintos modos en los bucles while. Estas habilidades te servirán en las próximas sesiones cuando programes tu proyecto.

**Aplica:** Pule tu cuestionario (página 95).

## 8 Vectores y refactorización

En esta sesión, los miembros del club adquirirán habilidades técnicas nuevas utilizando vectores y las pondrán en práctica para refactorizar su código.

**Aprende:** Echa un vistazo a la introducción a los vectores de «Aprende a programar 2».

Evaluación (página 99)

**Prueba:** Completa los rompecabezas del capítulo dedicado a los vectores de «Aprende a programar 2» (páginas 100-105).

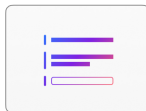
## 9 Añade opciones al cuestionario

Actualiza tu proyecto de cuestionario para incluir listas de opciones, y empieza a imaginar los proyectos que podrías crear con ellas.

**Aplica:** Añade opciones a tu cuestionario (página 106).

### Respuestas

- Texto
- Resumen de API
- Tipos



### Aprende a programar 2 Vectores

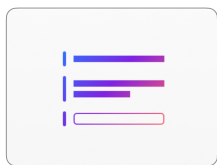
- Introducción
- Almacenar información
- Explorar las iteraciones
- Apilar bloques
- Organización
- Corregir errores de índice fuera de rango



### Respuestas

- Texto
- Resumen de API
- Tipos





# Pregunta a tus amigos

## 10 Diseña un proyecto nuevo

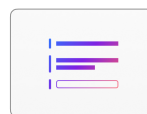
Piensa en qué otros proyectos podrías crear con el punto de inicio «Respuestas». En grupo, pon en común las ideas que se te hayan ocurrido con tus compañeros y, después, trabaja por tu cuenta para elegir una, identificar la finalidad y el público y esbozar un esquema.

## 11 Crea el proyecto

Crea tu proyecto con el punto de inicio «Respuestas». Sigue el esquema que has creado durante la sesión anterior.

### Respuestas

- Texto
- Resumen de API
- Tipos



## 12 Evalúa el proyecto

Prueba tu proyecto de área de juegos con tus compañeros. Ensaya cómo vas a explicar el funcionamiento de tu proyecto (y tus decisiones de diseño) en el evento público en el que compartirás tus creaciones.

### Respuestas

- Texto
- Resumen de API
- Tipos



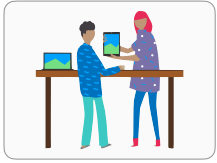
## Evento público

Celebra los logros del club en un evento público. Así podrás exponer tu proyecto, explicar el proceso de diseño y conocer la opinión de los demás.



# Diseña una app

```
func show(from actor: Actor) {guard state == .inactive else { return }state = .animatingToPicker// Use the idle for the c
commandSpeed = WorldConfiguration.Actor.idleSpeedoriginalWorldActor = actororiginalActorTransform =
scnNode.transformactor.reset()actor.scnNode.runAction(.playSoundEffect(tapSource)) // Reset the existing `pickerActors`.
d).for pickerActor in pickerActors {pickerActor.reset()pickerActor.isInCharacterPicker = true} let result = actor.perform
result.completionHandler = { _ inguard self.state == .animatingToPicker else TriggerDepthOfField(intro: true) { [unowned
adAndDisplayCharacters()}}}func triggerDepthOfField(intro: Bool, completion: CompletionBlock? = nil) {let root = scene?.r
ode = root?.childNode(withName: "camera", recursively: true),let camera = cameraNode.camera else { return } SCNTransaction
saction.animationDuration = CharacterPickerController.fadeDurationcamera.focusDistance = intro ? focusDistanceMax : 0came
de : fStopDefault SCNTransaction.completionBlock = completionSCNTransaction.commit()private func loadAndDisplayCharacters
return } / Setup overlay view & constraooverlayView.alpha = 0overlayView.autoresizingMask = [.flexibleWidth, .flexibleHeig
ldSubview(overlayView).view.bounds // Add a gesture recognizer to detect when character selection happens.let tapGesti
ction: #selector(selectCharacter(:)) overlayView.gestureRecognizers = [tapGesture // Setup overlay sc
```



# Diseña una app






## Resumen del módulo

En este módulo, los miembros del club trabajan en grupos reducidos para diseñar una app que ayude a resolver un problema de su entorno. Siguen un proceso de diseño que los lleva a intercambiar ideas, planificar su app, desarrollar un prototipo funcional en Keynote y evaluar el resultado.

El contenido del diseño se presenta en un [diario de diseño de apps](#) que ayuda a los miembros del club a anotar y dar seguimiento a sus ideas a medida que avanzan por el proceso. La idea es documentar el proceso para ayudar a reiterar y mejorar el proyecto. También sirve de referencia y de punto de partida para futuros proyectos.

Al final de este módulo, organiza una presentación de las apps para celebrar la imaginación y el ingenio de tu club. Descarga la [«Guía de presentación de apps»](#), donde encontrarás consejos y recursos para preparar el evento.

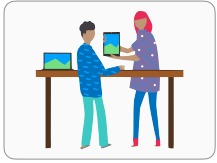
### Resumen de sesiones

-  Lluvia de ideas: 2 sesiones
-  Planificación: 2 sesiones
-  Prototipo: 5 sesiones
-  Evaluación: 3 sesiones
-  Presentación

## Recursos



Diario de diseño de apps



# Diseña una app

## 1–2 Lluvia de ideas

Explora ideas de apps para abordar algo que te importe y concéntrate en el objetivo y el público de la app.

### Lluvia de ideas

- Objetivo
- Ideas
- Prioridad



## 3–4 Planificación

Piensa en lo que hará tu app para lograr el objetivo mientras exploras las prácticas de inclusión y las prestaciones de iOS.

### Planificación

- Acciones del usuario
- Interacciones y estado de la app
- Elección de prestaciones
- Inclusión



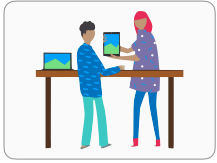
## 5–9 Prototipo

Diseña la interfaz de usuario de tu app, haz un guion gráfico con las pantallas y crea un prototipo funcional en Keynote.

### Prototipo

- Bocetos de pantallas
- Guion gráfico
- Mejora del comportamiento de la app
- Estilo del diseño
- Creación
- Icono y nombre de la app





# Diseña una app

## 10–12 Evaluación

Pon a prueba tu prototipo con compañeros y otras personas de la comunidad. Después haz iteraciones en función de los comentarios de los usuarios.

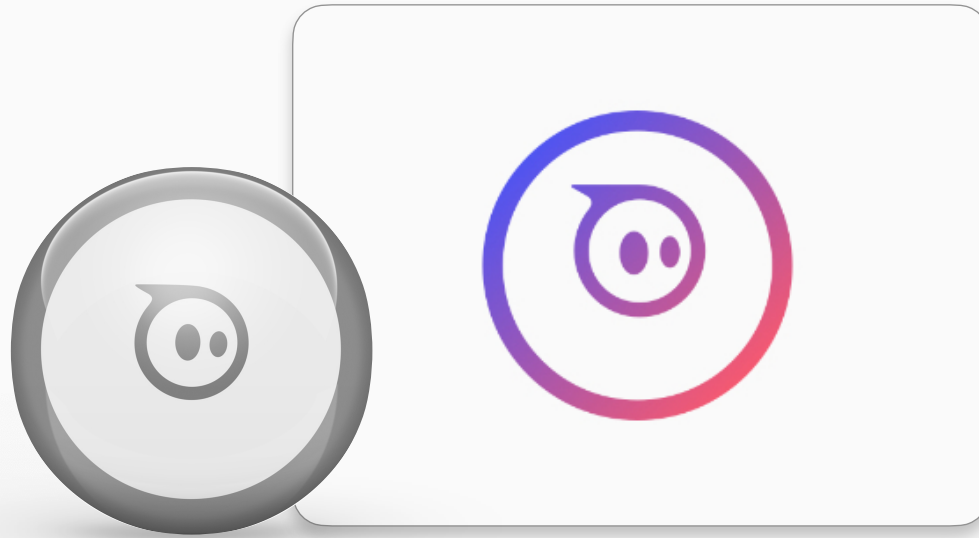
### Evaluación

- Presentación de la app
- Preparación
- Observación



## Presentación de apps

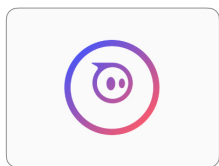
Organiza una presentación de apps para que los miembros del club expongan sus prototipos al público y expliquen la finalidad de sus apps. Inspírate a la hora de planificar y realizar el evento con la [«Guía de presentación de apps»](#).



# Haz un juego de Sphero

```
func show(from actor: Actor) {guard state == .inactive else { return }state = .animatingToPicker// Use the idle for the c
commandSpeed = WorldConfiguration.Actor.idleSpeedoriginalWorldActor = actororiginalActorTransform =
cnNode.transformactor.reset()actor.scnNode.runAction(.playSoundEffect(tapSource)) // Reset the existing `pickerActors`.
d).for pickerActor in pickerActors {pickerActor.reset()pickerActor.isInCharacterPicker = true} let result = actor.perform
result.completionHandler = { _ inguard self.state == .animatingToPicker else TriggerDepthOfField(intro: true) { [unowned
hadAndDisplayCharacters()}}}func triggerDepthOfField(intro: Bool, completion: CompletionBlock? = nil) {let root = scene?.r
ode = root?.childNode(withName: "camera", recursively: true),let camera = cameraNode.camera else { return } SCNTransaction
saction.animationDuration = CharacterPickerController.fadeDurationcamera.focusDistance = intro ? focusDistanceMax : 0came
de : fStopDefault SCNTransaction.completionBlock = completionSCNTransaction.commit()private func oadAndDisplayCharacters
return } // Setup overlay view & constraooverlayView.alpha = 0overlayView.autoresizingMask = [.flexibleWidth, .flexibleHeig
ldSubview(overlayView)view.bounds // Add a gesture recognizer to detect when character selection happens.let tapGesti
ction: #selector(selectCharacter(:)) overlayView.gestureRecognizers = [tapGesture // Setup overlay sc
```





# Haz un juego de Sphero

## Resumen del módulo

En este módulo, los miembros del club usan Swift Playgrounds para programar un robot Sphero y recrear los clásicos juegos recreativos. Los miembros del club necesitan al menos un robot Sphero por pareja.

Los miembros del club estudian los datos recopilados por el robot Sphero y cómo pueden usar estas prestaciones para crear juegos interactivos. Trabajan juntos para entender el código que se necesita para crear el juego y después lo editan para darle un toque diferente.

Los miembros del club ponen en práctica lo aprendido para diseñar su propio juego con uno o más robots Sphero. Exponen sus juegos en un evento público, invitan a los asistentes a ver el juego y a jugar con él, y explican sus decisiones de diseño y programación.

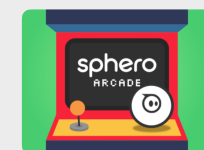
### Resumen de sesiones

- Sphero Pong: 3 sesiones
- Sphero Bop It: 2 sesiones
- Sphero Pac-Man: 2 sesiones
- Diseña un juego: 5 sesiones
- Evento público

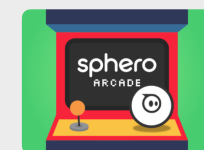
## Recursos



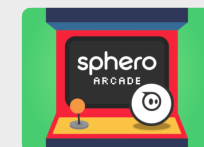
**Robot Sphero Mini**  
(uno por pareja)



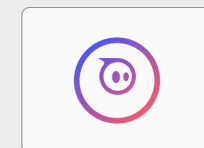
**El arcade de Sphero 1**



**El arcade de Sphero 2**



**El arcade de Sphero 3**



**Plantilla de Sphero**



# Haz un juego de Sphero

## 1 Sphero Pong

Echa un vistazo a «El arcade de Sphero 1» en Swift Playgrounds. Aprende a mover el Sphero, abre la página «Pong original» y juega en parejas. Determina el código que necesitas para crear el juego, haz un boceto de tus ideas y añade anotaciones con pseudocódigo.

### El arcade de Sphero 1

- Introducción
- Rodar
- Dirigir
- Orientación
- Impactos
- Pong original



## 2-3 Sphero Pong

En grupos reducidos, crea un juego de Sphero Pong «en directo» en el que uses los pies como raquetas. Al final de la tercera sesión, repasa el código que pensabas que ibas a necesitar para crear el juego Sphero Pong y debate si te ha hecho falta algo más.

### El arcade de Sphero 1

- Configuración del mundo real
- Ángulo de rebote
- Ida y vuelta
- Mantener el marcador
- Ganar el juego
- Jugar el juego



## 4-5 Sphero Bop It

Recrea por parejas el juego de Bop It con Sphero. Echa un vistazo a «El arcade de Sphero 2» para aprender a programar todos los gestos y aumentar la dificultad del juego. Edita el código para crear tus propios movimientos y determinar qué otro código podrías necesitar para vincular la interfaz del área de juegos al Sphero.

### El arcade de Sphero 2

- Introducción
- Tocar
- Lanzar
- Girar
- Agitar
- Aleatorizar el juego
- Rampa de dificultad
- Jugar el juego





# Haz un juego de Sphero

## 6–7 Sphero Pac-Man

Recrea por parejas el juego Pac-Man con el Sphero. Echa un vistazo a «El arcade de Sphero 3» para aprender a programar el Sphero como un joystick, calcular los puntos y crear enemigos. Edita el juego para aumentar la dificultad y determinar qué otro código podrías necesitar para crear todos los aspectos de la interfaz.

### El arcade de Sphero 3

- Introducción
- Controles simples
- Puntuación
- Poderes especiales
- Enemigos básicos
- Enemigo avanzado
- Jugar el juego



## 8–9 Sal del laberinto

Programa el Sphero para recorrer un laberinto en el área de juegos de la plantilla de Sphero. Diseña el laberinto y dibújalo con cinta adhesiva. Lo aconsejable es empezar con un laberinto sencillo. Puedes trabajar en pequeños grupos o crear un solo laberinto y recorrerlo con los robots Sphero para ver quién consigue salir antes y con más maña.

### Plantilla de Sphero

- Plantilla



## 10–12 Diseña un juego

Haz una lluvia de ideas para el diseño de tu juego con Sphero. El juego puede ser una versión física de un juego de recreativos, una carrera de obstáculos o un juego en el que participen dos o más robots Sphero. Esboza y planifica tu juego, y crea un proyecto de área de juegos con la plantilla de Sphero. Recuerda descomponer el juego en distintos elementos y usar los comentarios de tu código para compartir tus ideas.

### Plantilla de Sphero

- Plantilla



## Evento público

Celebra los logros del club en un evento público. Así podrás exponer tu proyecto, explicar el proceso de diseño y conocer la opinión de los demás.

