APRENDO A PROGRAMAR

PLAN DIFERENCIADO DE MATEMÁTICAS III Y IV MEDIO

Ideo Digital

kodea BHP Foundation

En el presente documento, se utilizan de manera inclusiva términos como “el docente”, “el estudiante”, “el profesor”, “el niño”, “el compañero” y sus respectivos plurales (así como otras palabras equivalentes en el contexto educativo) para referirse a hombres y mujeres. Esta opción obedece a que no existe acuerdo universal respecto de cómo aludir conjuntamente a ambos sexos en el idioma español, salvo usando “o/a”, “los/las” y otras similares, y ese tipo de fórmulas supone una saturación gráfica que puede dificultar la comprensión de la lectura.

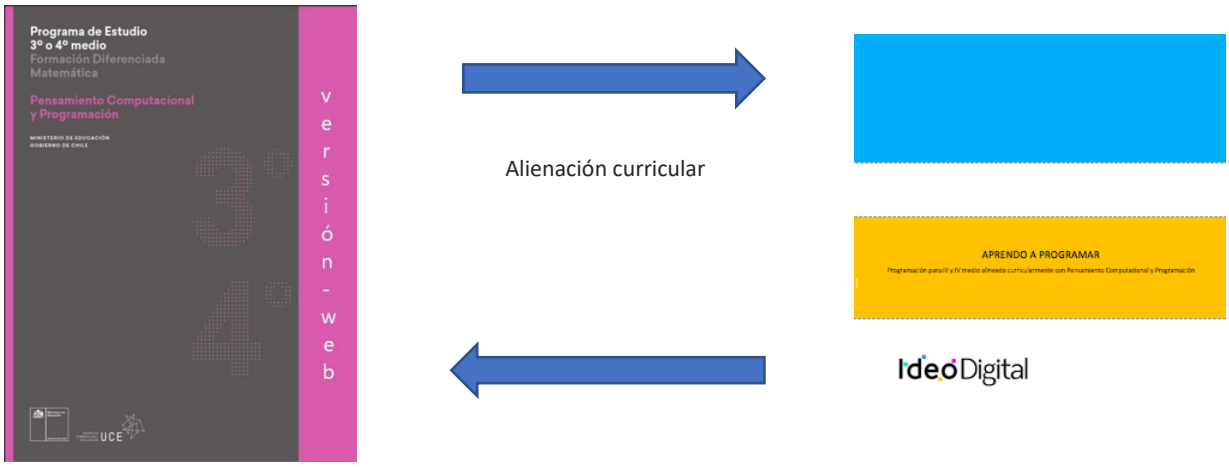
TABLA DE CONTENIDO

Presentación del programa

Aprendo a programar: programación integrada con Pensamiento computacional y programación pone el foco en relevar los Objetivos de Aprendizaje de las Bases Curriculares desde la lógica de la programación para los nivele de III y IV medio.

Para efectos pedagógicos significativos y coherentes con los intereses de los estudiantes, la integración de aprendizajes está enfocada en lecciones con y sin conexión, con la intención de incrementar instancias de aprendizaje que necesitan ser resueltas por medio de la programación, la interacción con equipos computacionales y el uso de materiales y recursos educativos que permiten al estudiantes avanzar desde ideas concretas o otras más abstractas. Asimismo, se busca una integración coherente con los conocimientos y habilidades propias del Pensamiento Computacional y Programación para estos niveles, como son los fundamentos de ciencias de la computación, el pensamiento crítico, la creatividad, la comunicación efectiva y la colaboración.

En cada una de las lecciones se favorece la adquisición de un lenguaje de programación que facilita la resolución de problemas en diferentes ambientes de enseñanza aprendizaje.



Propósito de aprendo a programar

El propósito de las lecciones es relevar estrategias didácticas asociadas al aprendizaje de la programación Yyel desarrollo del pensamiento técnico y tecnológico, que permiten abordar de manera simultánea los objetivos de aprendizaje prescritos en las Bases Curriculares vigentes para la asignatura de Pensamiento Computacional y Programación.

Esta propuesta de aprendizaje busca dar continuidad al Programa Fundamentos de la Computación de 1°básico a II año medio disponible en curriculum nacional, fortaleciendo conceptos y habilidades del pensamiento computacional, la resolución de problemas tecnológicos, el diseño sistemas y la comprensión del mundo a través de la tecnología, el ambiente y la sociedad.

Para el uso e implementación de las lecciones se sugieren propósitos, secuencias de aprendizaje, momentos de preparación, vocabulario y los objetivos de cada una, las cuales potencian e integran activamente los aprendizajes de los estudiantes a través de la resolución de problemas, el desarrollo de habilidades y la aplicación creativa de la programación, teniendo en cuenta el tiempo disponible y las particularidades de cada contexto escolar.

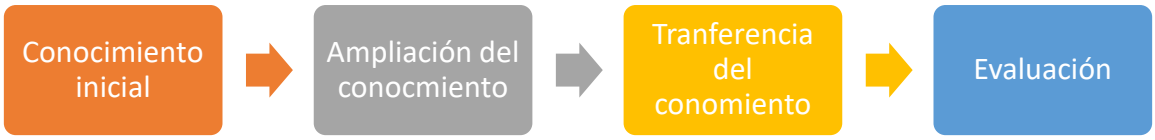
Las actividades consolidadas en cada lección se suman a una serie de experiencias de aprendizaje, con el objetivo de profundizar y afianzar el conocimiento de los contenidos vistos, así como también fortalecer las habilidades abordadas en cada unidad. Además, propone desafíos a los estudiantes, que los docentes podrán utilizar como ticket de salida y

parte de la evaluación de proceso, según la pertinencia y atingencia de los avances de sus estudiantes, ya que la idea es facilitar el proceso de enseñanza aprendizaje de los y las estudiantes.1

Cada lección será valorada, como una evaluación de proceso, con el fin de enriquecer los conocimientos adquiridos, a través de desafíos que serán considerados como una estrategia de evaluación de salida, retroalimentando de manera constante, así como también desarrollando la metacognición y metaevaluación de los estudiantes.

Aprendo a programar entrega una serie de sugerencias al docente, recomendaciones de recursos didácticos complementarios, como videos, tutoriales y bibliografía dispuesta tanto para profesores como para los y las estudiantes.2

Para poder abordar las lecciones, en cada una de ellas se sugiere la mencionada secuencia de aprendizaje, la cual está asociada a tiempos para cada instancia, siendo flexible de acuerdo con la asignación de horas de libre disposición que considere cada establecimiento:



Objetivos de Aprendizaje de Pensamiento computacional y programación3 que abordan las lecciones de Aprendo a Programar

Aprendo a programar contempla cuatro unidades, diseñadas de manera progresiva, considerando los Objetivos de Aprendizaje de Pensamiento computacional y programación.

Objetivos de Aprendizaje Pensamiento Computacional y programación 3° o 4° medio		Lecciones que cubren OA
OA 1.	Aplicar conceptos de Ciencias de la Computación – abstracción, organización lógica de datos, análisis de soluciones alternativas y generalización– al crear el código de una solución computacional.	Desde la lección 1 a la 25.
OA a.	Construir y evaluar estrategias de manera colaborativa al resolver problemas no rutinarios.	Desde la lección 5 a la 25.
OA d.	Argumentar, utilizando lenguaje simbólico y diferentes representaciones para justificar la veracidad o falsedad de una conjetura, y evaluar el alcance y los límites de los argumentos utilizados.	Desde la lección 5 a la 25.
OA g.	Elaborar representaciones, tanto en forma manual como digital, y justificar cómo una misma información puede ser utilizada según el tipo de representación.	Desde la lección 12 a la 25.

1 https://www.curriculumnacional.cl/portal/Diferenciado-Humanista-Cientifico/Matematica/Pensamiento-computacional-y-programacion/
2 https://www.curriculumnacional.cl/portal/Diferenciado-Humanista-Cientifico/Matematica/Pensamiento-computacional-y-programacion/
3 https://www.curriculumnacional.cl/portal/Diferenciado-Humanista-Cientifico/Matematica/Pensamiento-computacional-y-programacion/

OA k.	Analizar y evaluar el impacto de las tecnologías digitales en contextos sociales, económicos y culturales.	Desde la lección 5 a la 11.
Actitudes	Pensar con perseverancia y proactividad para encontrar soluciones innovadoras a los problemas.	Desde la lección 1 a la 25.
Actitudes	Interesarse por las posibilidades que ofrece la tecnología para el desarrollo intelectual, personal y social del individuo.	Desde la lección 5 a la 11.
Actitudes	Aprovechar las herramientas disponibles para aprender y resolver problemas.	Desde la lección 5 a la 25.

Estructura general del programa

El programa de estudio, Contempla cuatro unidades de aprendizaje, las cuales están divididas en actividades, cuya complejidad de los contenidos, estará estrechamente relacionado con la cantidad de actividades por unidad, asimismo, algunas actividades requieren conexión a internet (en línea) y otras se pueden realizar en el aula de clases sin conexión a internet.

UNIDAD 1: El Pensamiento Computacional e iniciación a la programación

Contenido 1: Introducción al Pensamiento Computacional

Lección1: Programación con papel cuadriculado	Sin conexión
Lección2: Introducción a los desafíos en línea	En línea
Lección3: Programación de relevos	Sin conexión
Lección4: Depuración del código	En línea

Contenido 2: Creación de animaciones

Lección5: Programación para el entretenimiento	En línea
Lección6: Trazar formas	En Línea
Lección7: Dibujar en Game Lab	En Línea
Lección8: Formas y aleatorio	En Línea
Lección9: Variables	En Línea
Lección10: Sprites	En Línea
Lección11: Ciclo de dibujo aleatorio	En Línea
Lección12: Patrón de contador desconectado	Sin conexión
Lección13: Movimiento de Sprite	En Línea
Lección14: Booleanos desconectados	En Línea
Lección15: Booleanos y condicionales	En Línea
Lección16: Condicionales y entrada del usuario	En Línea
Lección17: Otras formas de entrada	En línea
Lección18: Proyecto - Tarjeta interactiva	En Línea

Contenido 3: Creación de videojuegos

Lección19: Velocidad	En Línea
Lección20: Detección de colisión	En Línea
Lección21: Movimiento complejo de Sprite	En Línea
Lección22: Colisiones	En Línea
Lección23: Funciones	En Línea
Lección24: El proceso de diseño del juego	En Línea
Lección25: Uso del proceso de diseño del juego	En Línea

UNIDAD 1: ¿Qué es el pensamiento computacional y la programación?

Contenido 1: Introducción al pensamiento computacional

Lección 1: Programación con papel cuadriculado

Lección 2: Introducción a los desafíos en línea

Lección 3: Programación de relevos

Lección 4: Depuración del código

Contenido 2: Creación de animaciones

Lección 5: Diseñador de Minecraft

Lección 6: Trazar formas

Lección 7: Dibujar en Game Lab

Lección 8: Formas y aleatorio

Lección 9: Variables

Lección 10: Sprites

Lección 11: Ciclo de dibujo aleatorio

Lección 12: Patrón de contador desconectado

Lección 13: Movimiento de Sprite

Lección 14: Booleanos desconectados

Lección 15: Booleanos y condicionales

Lección 16: Condicionales y entrada del usuario

Lección 17: Otras formas de entrada

Lección 18: Proyecto - Tarjeta interactiva

Contenido 3: Creación de videojuegos

Lección 19: Velocidad

Lección 20: Detección de colisión

Lección 21: Movimiento complejo de Sprite

Lección 22: Colisiones

Lección 23: Funciones

Lección 24: El proceso de diseño del juego

Lección 25: Uso del proceso de diseño del juego

Unidad 1 - ¿Qué es el pensamiento computacional y la programación?

Contenido 1 - Introducción al pensamiento computacional

Resumen

- **Lección 1:** Programación con papel cuadriculado
- **Lección 2:** Introducción a los desafíos en línea
- **Lección 3:** Programación de relevos
- **Lección 4:** Depuración del código

Objetivos

- OA 1. Aplicar conceptos de Ciencias de la Computación –abstracción, organización lógica de datos, análisis de soluciones alternativas y generalización– al crear el código de una solución computacional.

Referencias

- Code Studio - Code.org
- Cuantrix - Fundación Televisa
- CSTA – Computer Science Teachers Association

Lección 1: Programación con papel cuadriculado

Lección sin conexión

Propósito

Al "programarse" unos a otros para hacer dibujos, los estudiantes tendrán la oportunidad de experimentar los conceptos clave de la programación de una forma divertida y accesible.
Al principio de la clase, los estudiantes usarán símbolos para instruirse entre ellos a pintar cuadrados en un papel cuadriculado, a fin de reproducir una imagen existente. Si dispone de tiempo, la Lección puede terminar con imágenes creadas por los mismos estudiantes.
Los objetivos de esta Lección son desarrollar habilidades de pensamiento crítico, sembrar interés por este curso e introducir algunos conceptos fundamentales de la programación que serán usados a lo largo del curso. Al introducir conceptos básicos, como secuenciación o algoritmos, a través de una Lección sin conexión, incluso los estudiantes que no se sienten familiarizados con un computador podrán sentar las bases para comprender estos temas. En esta Lección, los estudiantes aprenderán cómo desarrollar un algoritmo y codificarlo en un programa.

Secuencia para el aprendizaje

Conocimiento inicial (10 min)
Ampliación del conocimiento (30 min)
Transferencia del conocimiento (15 min)
Evaluación (10 min)

Objetivos

Los estudiantes serán capaces de:

- Restructurar una secuencia de pasos en un programa codificado.
- Explicar las limitaciones de traducir problemas desde el lenguaje humano al lenguaje de las máquinas
- para facilitar el proceso de desarrollo del programa.

Preparación

- (Opcional) vea el video Lección en acción en sección recursos para profesores.
- Imprima una guía de trabajo y una evaluación para cada estudiante.
- Asegúrese de que cada estudiante tenga su Bitácora.

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los Profesores:

- Programación con Papel Cuadriculado - Guía de respuestas de la guía de trabajo
- Programación con Papel Cuadriculado - Guía de respuestas de la evaluación
- Programación con Papel Cuadriculado - Video

- Para los estudiantes:
- Programación con Papel Cuadriculado - Evaluación
- Programación con Papel Cuadriculado - Ficha de actividades
- Programación con Papel Cuadriculado - Video sin conexión

Vocabulario

- **Algoritmo:** lista de pasos para realizar una tarea.
- **Programa:** algoritmo que ha sido codificado de forma que pueda ser ejecutado por una máquina

Estrategia de aprendizaje

Conocimiento inicial (10 min)

Introducción a la programación con papel cuadriculado

En esta Lección, los estudiantes codificarán instrucciones para guiarse unos a otros a hacer dibujos, sin que el resto del grupo vea la imagen original. Esta sección contextualizará el ejercicio para la clase.
Mostrar: vean uno de los videos a continuación para contextualizar a los estudiantes sobre la clase de cosas que puede hacer un robot:

- Robot Honda Asimo (1:51).
- Robot diseñador de huevos (3:15).
- Robot bailarín de Lego (1:35).

Análisis: ¿cómo creen que el robot sabe cómo hacer las cosas que hace? ¿los robots tienen un cerebro similar al nuestro?
Guíe el análisis hacia una conversación sobre cómo las personas programan a los robots para hacer cosas específicas, a través de comandos específicos

Este breve análisis tiene por objetivo hacer énfasis en que los robots, a pesar de que pareciesen comportarse como humanos, realmente sólo responden a su programa. Es probable que los estudiantes hagan referencia a algunos robots de las películas y de la televisión con comportamientos más humanos. Guíelos a considerar robots que hayan visto u oído en la vida real, como el Roombas, o incluso asistentes digitales como Alexa o el asistente de Google.

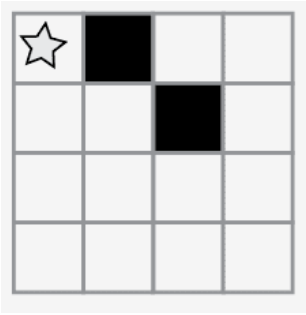
Ampliación del conocimiento (30 min)

Práctica en conjunto

En esta Lección, los estudiantes tomarán los roles de programador y robot. En una hoja de papel cuadriculado, pintarán cuadrados de acuerdo con los programas que se hayan escrito los unos a los otros.
Distribuir: los estudiantes usarán una cuadrícula de 4x4 (u hojas de papel cuadriculado seccionado en cuadriculas de 4x4). También necesitarán una imagen de modelo.
Mostrar: proyecte estos comandos o escríbalos en la pizarra. No estarán mucho tiempo ahí, sólo lo suficiente para ayudar a los estudiantes a hacer la transición de algoritmos a programas.

```
Moverse un cuadrado a la derecha
Moverse un cuadrado a la izquierda
Moverse un cuadrado arriba
Moverse un cuadrado abajo
Rellenar el cuadrado
```

Generar una instancia de aprendizaje donde los programadores serán los estudiantes y el docente sigue las instrucciones al pie de la letra (como si fuese un robot). Luego, dividimos en grupos para que todos tengan su turno.
Modelar: muestre la imagen que usará de ejemplo y la cuadrícula en blanco que rellenará con su Sistema de Ejecución Automática (SEA). Asegúrese de que las instrucciones, la cuadrícula y la imagen permanezcan visibles al mismo tiempo.



Comentarios
Presentar un robot imaginario que funciona con un Sistema de Ejecución Automática (SEA). Esto significa que reaccionaré de forma automática a sus instrucciones, pero sólo a las que pueda entender.
Empiecen en la esquina superior izquierda. Guíen mi SEA diciéndome las instrucciones en voz alta.
Modelar: a continuación, la clase podría darle instrucciones como éstas. Cuando escuche una instrucción que pretenda seguir, asegúrese de repetir dicha instrucción en voz alta, de manera que los estudiantes puedan llevar un registro de sus movimientos.

```
Moverse un cuadrado a la derecha
Rellenar el cuadrado
Moverse un cuadrado a la derecha
Moverse un cuadrado abajo
Rellenar el cuadrado
```

Continúe con la Lección hasta completar la cuadrícula de ejemplo.
Captar: escriba cada uno de los comandos, de manera que los estudiantes puedan ver todos los pasos realizados para dibujar la imagen.

```
Moverse un cuadrado a la derecha
Rellenar el cuadrado
Moverse un cuadrado a la derecha
Moverse un cuadrado abajo
Rellenar el cuadrado
```

Recordar la definición de algoritmos. Como programadores podemos entenderlos fácilmente. PERO ¿qué pasa si queremos escribir el algoritmo para un dibujo como este?
Mostrar: muestre una imagen más complicada, como ésta.

A continuación, comience a escribir algunas de las instrucciones para replicar la imagen. Con suerte, los estudiantes verán que escribir todo a mano podría fácilmente volverse una pesadilla.

Mostrar: muestre esta lista de símbolos.

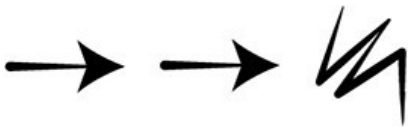
```
Moverse un cuadrado a la derecha
Rellenar el cuadrado
Moverse un cuadrado a la derecha
Moverse un cuadrado a la derecha
Rellenar el cuadrado
Moverse un cuadrado abajo
Moverse un cuadrado a la izquierda
Rellenar el cuadrado
Moverse un cuadrado a la izquierda
Moverse un cuadrado a la izquierda
Rellenar el cuadrado
¡¡Y 12 instrucciones más!!
```

El objetivo de este análisis es llegar a la idea de que los estudiantes pueden usar símbolos para representar frases completas. Una vez que lo comprendan, coménteles que pasar de enlistar pasos detallados a codificarlos, se llama "programación".

Análisis: ¿cómo podemos usar estos símbolos para facilitar nuestras instrucciones?

Continúe explorando ideas que apoyen la transición de instrucciones verbales a símbolos. Una vez que los estudiantes comprendan la idea, indique que el texto:

"Moverse un cuadrado a la derecha, moverse un cuadrado a la derecha, rellenar el cuadrado" Ahora corresponde al programa:

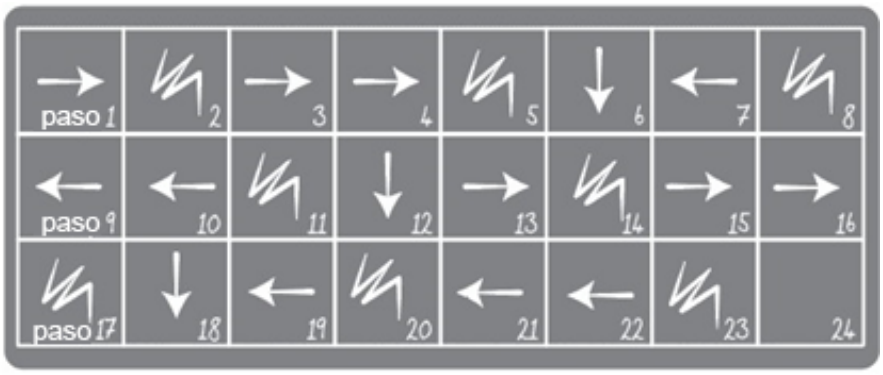


Modelar: ahora, que la clase le ayude a dibujar la imagen grande usando sólo símbolos. Por ahora, no se preocupe si se incluyen pasos innecesarios; si el programa funciona para recrear la imagen, considérelo correcto.

En este punto, los estudiantes podrían estar emocionados y animados dando sugerencias. Si entienden lo esencial del ejercicio, este es un buen momento para analizar otras alternativas para rellenar la misma cuadrícula. Si aún no están listos, guarde esta idea para otro día y realice otro ejemplo.

Vea una solución de ejemplo a continuación:

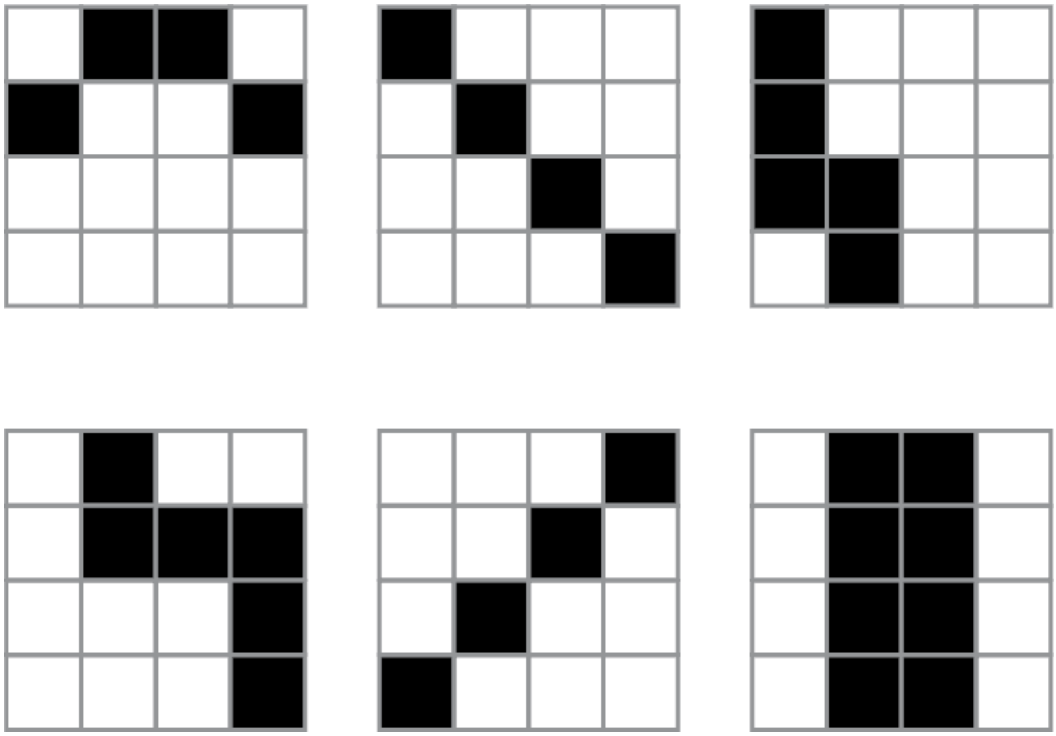
Note como hemos escrito el programa de izquierda a derecha, como se leería un libro en español. Algunos estudiantes prefieren este método, mientras que otros prefieren empezar cada línea de la cuadrícula en una nueva línea de la hoja. La forma en que escriban el programa no importa mucho, mientras los demás estudiantes puedan leer y seguir el programa.



Turno del estudiante

Grupo: divida a los estudiantes en parejas o pequeños grupos.

- Que cada grupo/pareja elija una imagen de la guía de trabajo.
- Discutan con sus compañeros el algoritmo necesario para dibujar la imagen elegida.
- Conviertan el algoritmo en un programa, usando los símbolos.
- Intercambien los programas con otros grupos/parejas, para que dibujen las imágenes de otros.
- Elijan otra imagen y ¡volvamos a empezar!



Transferencia del conocimiento (15 min)

Escribir en la bitácora y charla rápida

El acto de escribir en su bitácora sobre lo aprendido, respecto de si les pareció útil y de lo que sintieron, ayuda a sus estudiantes a fortalecer cualquier conocimiento que hayan obtenido hoy y servir como un resumen al que puedan recurrir en el futuro.

Sugerencias para la bitácora:

- ¿Sobre qué se trataba la Lección de hoy?
- ¿Cómo te sentiste durante la Lección?
- Dibuja otra imagen que podrías codificar. ¿Puedes escribir el programa que corresponde con la imagen?
- ¿Qué otro tipo de robots podríamos programar si cambiáramos lo que significan las flechas?

Evaluación (10 min)

- Entregue la Evaluación – Programación con papel cuadriculado. Luego de haber explicado claramente las instrucciones, permita que los estudiantes realicen la evaluación de forma individual.
- Gracias a la Lección previa, esto no debería significar ningún problema para ellos.

Experiencias de aprendizaje de profundización

Use estos Contenidos para ampliar el aprendizaje de los estudiantes. Se pueden usar como Contenidos extras fuera del aula.

Cada vez mejor

- Que su clase intente hacer sus propias imágenes.
- ¿Pueden descifrar cómo codificar las imágenes que acaban de crear?

Desafío de curso

- Dibuje una imagen en una cuadrícula de 5x5 en lugar de una 4x4.

Lección 2: Introducción a los desafíos en línea

Lección en línea Ver en Code Studio

Propósito

En este conjunto de desafíos, los estudiantes comenzarán con una introducción (o repaso, dependiendo de la experiencia de su clase) del espacio de trabajo en línea de Code.org. Habrá videos indicando las funciones básicas del espacio de trabajo, como los botones ejecutar (run), reiniciar (reset) y paso (step). En estos videos también se aborda cómo arrastrar, borrar y conectar bloques Blockly. Luego, sus estudiantes pondrán en práctica sus habilidades de secuenciación y depuración en un laberinto. A partir de ahí, los estudiantes verán nuevos tipos de desafíos, como la recolectora, el artista o la cosechadora, mientras aprenden lo básico de los bucles.
Comprendemos que en cada sala de clases hay un espectro de comprensión diferente para cada tema. Algunos estudiantes pueden ser muy hábiles con los computadores mientras que otros pueden tener muy poca experiencia con ellos. Con el objetivo de nivelar el área de juego (y de aprendizaje), hemos desarrollado esta "etapa de reforzamiento" para el curso D. Esto puede ser tanto una introducción como un repaso de cómo usar Code.org y conceptos básicos de las Ciencias de la Computación.

Secuencia para el aprendizaje

Conocimiento inicial (10 min)
Lección puente: programación (10 min)
Ampliación del conocimiento (30 min)
Transferencia del conocimiento (10 min)

Objetivos

Los estudiantes serán capaces de:

- Ordenar comandos de movimientos como pasos secuenciales de un programa.
- Modificar un programa existente para reparar errores.
- Descomponer una secuencia larga de instrucciones en una secuencia de repetición más corta.

Preparación

- Realice los desafíos para encontrar cualquier área potencialmente problemática para su clase.
- Asegúrese de que cada estudiante tenga su Bitácora.

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los estudiantes:

- Programación en Parejas - Video

Vocabulario

- **Bug (error)**: parte de un programa que no funciona correctamente.
- **Depurar (debugging)**: encontrar y solucionar los problemas en un algoritmo o programa.
- **Bucle (loop):** la acción de hacer algo una y otra vez.
- **Programa**: algoritmo que ha sido codificado de forma que pueda ser ejecutado por una máquina.
- **Programación**: el arte de crear un programa.

Estrategia de aprendizaje

Conocimiento inicial (10 min)

Introducción

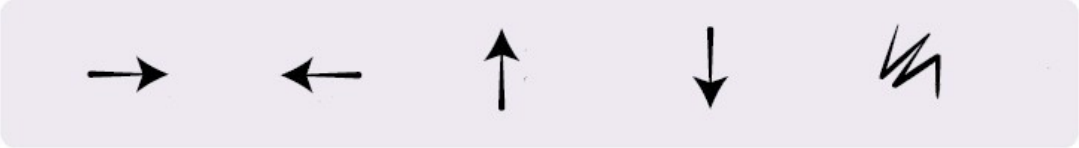
Los estudiantes aprenderán muchos conceptos nuevos o repasarán muchos conceptos básicos. En base a la experiencia de su clase, puede revisar las palabras programa, programación, bug, depurar, ciclo o bucle en el glosario del curso o pasar directamente a la Lección puente. Si las definiciones no están cubiertas explícitamente, recomendamos usar las palabras en oraciones.

Lección puente: programación (10 min)

Esta Lección ayudará a sus estudiantes a traer los conceptos sin conexión de "programación con papel cuadriculado" al mundo digital al cual se están sumergiendo.

Revisión de desafíos en línea en conjunto

Elija un desafío de la Lección. Para esta Lección recomendamos progresar con estos desafíos. Divida a los estudiantes en grupos de tres o cuatro. Pídales que programen a "Red", de Angry Birds, para que llegue al cerdo, usando las flechas de "programación con papel cuadriculado".
La clase no necesitará usar el último símbolo.



Una vez que toda la clase tenga una respuesta, discutan el camino a seguir en conjunto como clase.

Ampliación del conocimiento (30 min)

Desafíos en línea

Los profesores son de vital importancia en la educación de las Ciencias de la Computación y juegan un rol fundamental para producir un ambiente vibrante y colaborativo en la sala de clases. Durante las actividades en línea, el rol del profesor es alentar y apoyar. Los desafíos en línea están estructurados para estar centrados en el estudiante, así que los profesores deben evitar involucrarse cuando los alumnos tengan problemas para resolverlos. Algunas ideas de cómo hacerlo son:

Enséñeles a los estudiantes la manera correcta de trabajar en parejas:

- No usar el teclado del compañero.
- No tocar el *mouse* del compañero.
- Asegúrate que tu compañero pueda describirte la solución en voz alta antes de que vayas.

- Use la programación en parejas cada vez que sea posible durante la Lección.
- A través de desafíos o preguntas, anime a los estudiantes para que busquen respuestas con sus respectivas parejas.
- Las preguntas sin responder pueden ser delegadas a un grupo cercano, que podría ya tener la respuesta.
- Recuérdeles usar el proceso de depuración antes de que usted se acerque a ayudar.
- Pida a los estudiantes que describan el problema que estén viendo. ¿Qué se supone que debe hacer?, ¿qué hace?, ¿qué te dice eso?
- Recuérdeles que la frustración es un paso en el camino del aprendizaje y que la perseverancia dará sus frutos.
- Si un estudiante sigue con problemas para avanzar después de todo esto, haga preguntas clave para que los estudiantes identifiquen el bug (error) por ellos mismos.

Lección en Code Studio

Acceden a la plataforma para realizar los desafíos en línea.

Transferencia del conocimiento (10 min)

Escribir en la bitácora

El acto de escribir en su bitácora sobre lo aprendido, respecto de si les pareció útil y de lo que sintieron, ayuda a sus estudiantes a fortalecer cualquier conocimiento que hayan obtenido hoy y servir como un resumen al que puedan recurrir en el futuro.
Sugerencias para la bitácora:

- ¿Sobre qué se trataba la Lección de hoy?
- ¿Cómo te sentiste durante la Lección?
- Haz una lista de los bugs que encontraste en tus programas hoy.
- ¿Cuál fue tu desafío favorito? Dibuja a tu personaje favorito completando desafíos.

Sugerencias de Evaluación

Se sugiere evaluar formativamente los aprendizajes:

- Resuelven problemas
- Asignan roles en los grupos
- Dividen un problema en otros menores

Lección 3: Programación de relevos

Lección sin conexión

Propósito

Esta Lección comenzará con una breve Lección de depuración y perseverancia. Luego, rápidamente se transformará en una carrera contra el tiempo, mientras los estudiantes se dividen en grupos y trabajan en equipo para escribir un programa, una instrucción a la vez.
El trabajo en equipo es muy importante en la computación. Los equipos escriben y depuran códigos en conjunto, en lugar de trabajar de forma individual. Los estudiantes aprenderán a trabajar en equipo intentando ser lo más eficientes posible.
Esta Lección también involucra el concepto de urgencia, lo cual enseñará a los estudiantes que deben administrar su tiempo cuidadosamente, evitando cometer errores sin comprometer mucho tiempo. Esta experiencia puede ser estresante (¡y lo será!). Asegúrese de compartir con los estudiantes las herramientas para lidiar con la frustración.

Secuencia para el aprendizaje

Conocimiento inicial (15 min)
Ampliación del conocimiento (20 min)
Transferencia del conocimiento (15 min)

Objetivos

Los estudiantes serán capaces de:

- Definir ideas usando código y símbolos.
- Verificar el trabajo hecho por compañeros.
- Identificar las señales de la frustración.

Preparación

- Vea el Video – Programación de relevos.
- Encuentre un espacio abierto para esta Lección, como el gimnasio o un área verde.
- Imprima un Paquete de actividades – Programación de relevos para cada grupo.
- Provea a cada grupo hojas y lápices.
- Imprima una Guía de trabajo – Programación de relevos para cada estudiante.
- Asegúrese de que cada estudiante tenga su Bitácora.

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los profesores:

- Programación de relevos - Guía de respuestas de la guía de trabajo
- Programación de relevos - Imagen de depuración para el docente
- Programación de relevos – Video

Para los estudiantes:

- Programación de relevos - Paquete de actividades
- Programación de relevos - Guía de trabajo
- Programación de relevos – Video

Vocabulario

- **Algoritmo:** lista de pasos para realizar una tarea.
- **Bug (error)**: parte de un programa que no funciona correctamente.
- **Depurar (debugging)**: encontrar y solucionar los problemas en un algoritmo o programa.
- **Frustración**: sentirse molesto o enojado porque algo no funciona como quieres.
- **Perseverancia**: intentarlo una y otra vez, incluso cuando algo es muy difícil.
- **Programa**: algoritmo que ha sido codificado de forma que pueda ser ejecutado por una máquina.

Estrategia de aprendizaje

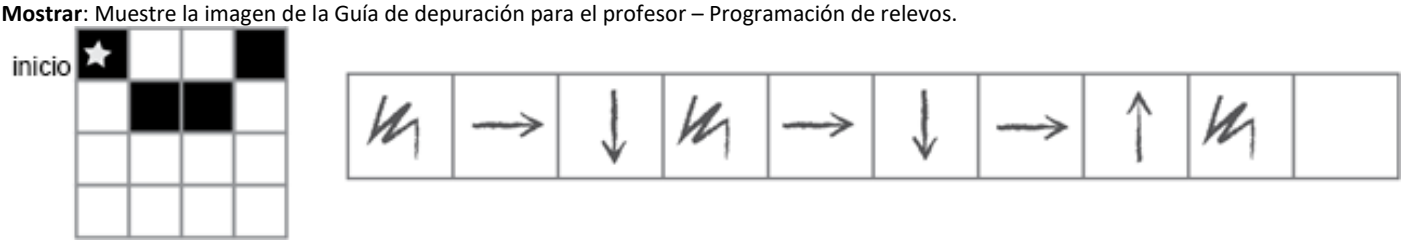
Conocimiento inicial (15 min)

Recordemos que en "programación con papel cuadriculado" guiamos el Sistema de Ejecución Automática (SEA) de nuestros compañeros usando flechas. Esta sección trae de vuelta esas ideas, las cuales serán necesarias en la Ampliación del conocimiento.

¿Dónde me equivoqué?

Objetivo: en esta Lección, queremos ayudar a los estudiantes a identificar y resolver bugs en sus programas. La forma más sencilla de hacerlo es trabajar en un programa que ya contenga bugs, pero que no sean culpa de los estudiantes. Una vez que le hayan ayudado a reparar "su" programa, comparta con ellos lo frustrante que puede llegar a ser cometer errores, y ayúdeles a ver que esos sentimientos son absolutamente normales, no deben sentirse avergonzados por ellos.

Mostrar: Muestre la imagen de la Guía de depuración para el profesor – Programación de relevos.



Análisis: Presente la imagen e indique que tiene un bug ¿pueden identificarlo?
Tómese un momento para repasar las reglas: comenzar en la estrella, seguir las instrucciones paso a paso, terminar cuando se hayan ejecutado todos los cuadrados de la derecha.

Reflexión: ¿pueden descifrar por qué el programa no funciona?
Interacción: pida a los estudiantes que trabajen en equipo para ver si pueden descifrar cómo debería estar escrito el programa.
Exposición: pregunte si alguien encontró una manera de resolver el problema. Cuando le den una respuesta correcta, recuerde la definición de depuración

Discusión: En la programación muchas veces nos vamos a ver enfrentados a la frustración, donde quizás te veas tentado a rendirte; sin embargo, la frustración es un sentimiento normal y es un gran indicio de que estás a punto de aprender algo. En lugar de rendirte, intenta ser perseverante. Sigue intentándolo una y otra vez. Después de algunos intentos, ¡empezarás a entender cómo depurar tus problemas!

Distribuir: para asegurarse de que los estudiantes entienden la idea de encontrar y solucionar problemas (depurar), distribuya la Guía de trabajo – programación de relevos y que trabajen en parejas.

Sugerencias para lidiar con la frustración:
- Contar hasta 10
- Respirar hondo
- Escribir acerca del problema
- Hablar con algún amigo sobre el problema
- Pedir ayuda

Sugerencias para ser perseverante:
- Llevar registro de lo que ya has intentado
- Describir qué es lo que pasa
- Describir qué debería hacer
- ¿Qué te dice eso?
- Haz un cambio y vuelve a intentar

Opcional: Si no quiere invertir mucho tiempo en esta Lección, pueden realizarla en conjunto como clase.
Transición: ¡llegó la hora de la acción!

Ampliación del conocimiento (20 min)

Programación de relevos

Con los conceptos de "programación con papel cuadriculado" en mente, ¡es momento de dividirse en equipos y prepararse para la Lección en relevos!
Preparación: prepárese imprimiendo el Paquete de actividades – Programación de relevos para cada equipo, de 4 a 5 estudiantes. Corte o doble cada página por la línea punteada.
Revise las reglas del juego con su clase:

- Dividirse en equipos de 4-5 estudiantes.
- Que cada grupo haga una fila, uno detrás de otro.
- Colocar una de las imágenes guía al otro lado de la sala/gimnasio/área para cada equipo (use la misma imagen para todos los equipos).
- El primer estudiante de la fila debe correr hasta la imagen, revisarla, y escribir el primer símbolo del programa para reproducir esa imagen.
- El primer estudiante debe volver y tocar al siguiente de la fila. Luego, debe ponerse al final de la fila.
- La siguiente persona en la fila debe correr hasta la imagen y revisarla. Luego, debe revisar el programa que se ha escrito y realizar sólo una de estas dos opciones: depurar el programa tachando los símbolos incorrectos o agregar un símbolo nuevo. El estudiante luego corre de vuelta y toca al siguiente en la fila para que sea su turno.
- Repetir el punto anterior hasta que un equipo termine el programa.

Algunas cosas que deben ser aclaradas y recordadas de vez en cuando:

- Sólo una persona de cada grupo puede estar frente a la imagen a la vez.
- Está permitido discutir el algoritmo con el resto del grupo mientras están en la fila, incluso pueden planificar quién y qué van a escribir en el programa.
- Cuando un estudiante depura el programa tachando una instrucción incorrecta (o un conjunto de instrucciones incorrectas), cuenta como su turno. El siguiente estudiante debe descifrar cómo corregir las instrucciones eliminadas.

¡El primer grupo que consiga un programa que calce con la imagen es el ganador! Repita esta Lección varias veces, aumentando la dificultad. Repita el juego tantas veces como pueda, hasta que se agote el tiempo o hasta que los estudiantes se sientan fatigados.
Transición: una vez que el juego acabe, reúna a todos los estudiantes en un círculo para compartir lo aprendido.
Discusión: ¿qué aprendimos hoy?

- ¿Qué tal si cada persona en la fila pudiese hacer 5 flechas a la vez?
- ¿Qué tan importante puede ser depurar nuestro propio trabajo y el trabajo del programador anterior a nosotros?
- ¿Y si pudiesen hacerse 10 flechas?
- ¿Y si fuesen 10.000? ¿Sería más o menos importante?
- ¿Crees que un programa es mejor o peor cuando más de una persona trabaja en él?
- ¿Crees que las personas comenten más o menos errores cuando están apurados?
- Si encuentras un error ¿debes deshacer todo el programa y comenzar desde cero?

Transferencia del conocimiento (15 min)

Escribir en la bitácora

El acto de escribir en su bitácora sobre lo aprendido, respecto de si les pareció útil y de lo que sintieron, ayuda a sus estudiantes a fortalecer cualquier conocimiento que hayan obtenido hoy y servir como un resumen al que puedan recurrir en el futuro.
Sugerencias para la bitácora:

- ¿Sobre qué se trataba la Lección de hoy?
- ¿Cómo te sentiste durante la Lección?
- ¿Cómo es que el trabajo en equipo influyó en el éxito de escribir el programa de hoy?
- ¿Te sentiste frustrado en algún punto? ¿qué hiciste respecto a eso?

Experiencias de aprendizaje de profundización

Use estos contenidos para ampliar el aprendizaje de los estudiantes. Se pueden usar como Contenidos extras fuera del aula.

Pasa la hoja

- Si no tiene tiempo o espacio para una Lección de relevos al aire libre, puede reunir a los estudiantes en grupos y que pasen la hoja con el programa de un puesto a otro. Que cada estudiante haga una flecha antes de pasar la hoja a un compañero.

Llénalo, muévelo

- Cómo profesor, dibuje una imagen con tantos cuadrados pintados como niños haya en cada grupo.

- Pida a los estudiantes que anoten tantas flechas en el programa como sea necesario para llegar a un cuadrado pintado, incluyendo la acción de pintar ese cuadrado, antes de pasar la hoja al siguiente estudiante.

Depurando juntos

Dibuje una imagen en la pizarra. Pídale a cada estudiante crear un programa para esa imagen. Luego, que intercambien sus programas con sus compañeros de al lado y depuren sus códigos.

- Encierra en un círculo el primer paso incorrecto, luego devuelve el programa.
- Deles a los estudiantes otra oportunidad para revisar y depurar sus propios programas.
- Pida que un voluntario comparta su programa.
- Pregunte:
- ¿Cuántos tienen el mismo programa?
- ¿Alguien tiene algo diferente?

Sugerencias de evaluación

Se sugiere el siguiente indicador para evaluar formativamente los aprendizajes:

- Identifican las partes que componen una situación.
- Elaboran instrucciones.

Lección4: Depuración con Laurel

Lección en línea Ver en Code Studio

Propósito

En esta Lección en línea, los estudiantes practicarán cómo depurar en un ambiente de "recolección", al practicar cómo leer y editar códigos para reparar desafíos con algoritmos simples, bucles y bucles anidados.
El propósito de esta Lección es enseñar a los estudiantes que fallar es normal cuando se estamos aprendiendo nuevas habilidades. A los estudiantes se les darán programas que NO funcionan correctamente y se les pedirá repararlos. Este proceso, llamado "depuración", les ayudará a desarrollar habilidades de pensamiento crítico y resolución de problemas, habilidades que los acompañarán mientras avanzan a proyectos de programación cada vez más complejos.

Secuencia para el aprendizaje

Conocimiento inicial (15 min)
Lección puente: depuración (15 min)
Ampliación del conocimiento (30 min)
Transferencia del conocimiento (15 min)

Objetivos

Los estudiantes serán capaces de:

- Leer y comprender un código dado.
- Identificar un bug y los problemas que causa en un programa.
- Describir e implementar un plan para depurar un programa.

Preparación

- Realice los desafíos para encontrar cualquier área potencialmente problemática para su clase.
- Asegúrese de que cada estudiante tenga su Bitácora.

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los estudiantes:

- Bloques sin conexión (cursos C-F) – Manipulativos
- Programación de relevos - Paquete de actividades

Vocabulario

- **Bug (error)**: parte de un programa que no funciona correctamente.
- **Depurar (debugging)**: encontrar y solucionar los problemas en un algoritmo o programa.

Estrategia de aprendizaje

Conocimiento inicial (15 min)

Introducción

Una de las partes más importantes en el aprendizaje de la programación es aprender a depurar. Pregunte a la clase si alguna vez han aprendido una habilidad nueva y si alguna vez han fallado.
Por ejemplo:

- Aprender a andar en bicicleta y caerse.
- Aprender a hornear y quemar la comida.
- Aprender algún deporte y no ganar ningún juego.

Equivocarse es muy común cuando se aprenden cosas nuevas. Pida a los estudiantes que discutan sobre situaciones en que se hayan equivocado y cómo las superaron.
En la programación, los programadores normalmente encuentran “bugs” en sus códigos.

- Bug (error): parte de un programa que no funciona correctamente.
- Un bug puede realmente arruinar un programa, por lo que es importante aprender a “depurar” el código.
- Depurar (debugging): encontrar y solucionar los problemas en un algoritmo o programa.
- Entretenga la conversación si cree que su clase necesita una mayor introducción, pero no olvide dejar tiempo para una de las actividades puente.

Lección puente: depuración (15 min)

Estas actividades ayudarán a sus estudiantes a traer los conceptos sin conexión de “depuración sin conexión: programación de relevos” al mundo digital al cual se están sumergiendo. Elija una de las siguientes actividades:

Lección sin conexión: bloques de papel

Divida a la clase en equipos de 3 a 5 estudiantes y encuentre un lugar amplio, puede ser un gimnasio o un área verde. Ordene los equipos en fila, similar a como se hizo en la Lección “programación de relevos”. Tome un diseño mediadamente difícil del Paquete de actividades – Programación de relevos. A una distancia considerable, coloque una imagen por cada equipo. Además, entrégueles suficientes bloques de papel de los Manipulativos – Bloques sin conexión (cursos C - F). Cada equipo necesitará muchos bloques llena 1 y mover____. Los bloques mover____ pueden ser rellenados con anticipación o durante el mismo juego. De cualquier modo, asegúrese de que los bloques estén bien definidos mientras se desarrolle el juego.
Una vez que todos los equipos estén formados, muestre o lea en voz alta las reglas:

- El primer estudiante en la fila debe correr hacia la imagen, revisarla, y colocar el primer bloque de código en el programa para reproducir esa imagen.
- El primer estudiante debe volver y tocar al siguiente de la fila. Luego, debe ponerse al final de la fila.
- La siguiente persona en la fila debe correr hasta la imagen y revisarla. Luego, debe revisar el programa que se ha escrito y hacer una de dos: depurar el programa quitando los bloques de código incorrectos o agregar un bloque nuevo. Luego, el estudiante debe correr de vuelta y toca al siguiente en la fila para que sea su turno.
- Repetir el punto anterior hasta que un equipo termine el programa.

Asegúrese de que los estudiantes usen sólo los bloques llena 1 y mover____ y que sólo coloquen un bloque por turno. ¡El primer equipo que escriba correctamente el código de su imagen, gana!

Revisión de desafíos en línea en conjunto

Agrupe a los estudiantes en grupos de 3. Elija un desafío de la Lección, recomendamos el número 5. Pídales que se sienten frente a un computador con el desafío en pantalla. Cada equipo puede tener sólo un computador y un estudiante mirando la pantalla. Muestre o lea las reglas:

- Sólo un estudiante por equipo puede estar mirando la pantalla.
- Esa persona sólo puede añadir o borrar un bloque a la vez. Cuando esa persona haya añadido o eliminado un bloque, puede tocar el hombro de un compañero para que sea su turno.
- El siguiente estudiante puede ponerse frente a la pantalla y jugar su turno.
- Los turnos no pueden repetirse ni saltarse, todos deben jugar la misma cantidad de veces.

¡El primer equipo en resolver el desafío correctamente, gana!

Ampliación del conocimiento (30 min)

Desafíos en línea

Podría ser muy útil que los integrantes de los equipos de la Lección puente se sienten cerca unos de otros. Cada estudiante debe realizar los desafíos de forma individual o en parejas; sin embargo, contar con un grupo de trabajo conocido para hacer y responder preguntas puede ayudarlos a sentir más confianza y a comprender mejor el tema.

Lección en Code Studio

Acceder a Code Studio para realizar los desafíos de depuración.

Transferencia del conocimiento (15 min)

Escribir en la bitácora

El acto de escribir en su bitácora sobre lo aprendido, respecto de si les pareció útil y de lo que sintieron, ayuda a sus estudiantes a fortalecer cualquier conocimiento que hayan obtenido hoy y servir como un resumen al que puedan recurrir en el futuro.

Sugerencias para la bitácora:

- ¿Sobre qué se trataba la Lección de hoy?
- ¿Cómo te sentiste durante la Lección?
- ¿Qué es un bug? ¿cómo sabes que hay un bug en tu programa?
- ¿Qué significa “depurar” un código? ¿cómo depuras un programa?

Sugerencia para evaluación

Se sugiere el siguiente indicador para evaluar formativamente los aprendizajes:

- Comprenden y analizan un problema.
- Piensan en la solución.
- Buscan alternativas; es decir, preparan los elementos para que desarrollen el programa y resuelvan el problema.

Unidad 1 - ¿Qué es el pensamiento computacional y la programación?

Contenido 2 - Creación de animaciones

Resumen

- **Lección 5:** Programación para el entretenimiento
- **Lección 6**: Trazar formas
- **Lección 7:** Dibujar en Game Lab
- **Lección 8:** Formas y aleatorio
- **Lección 9**: Variables
- Lección 10: Sprites
- **Lección 11:** Ciclo de dibujo aleatorio
- **Lección 12:** Patrón de contador desconectado
- **Lección 13:** Movimiento de Sprite
- **Lección 14:** Booleanos desconectados
- **Lección 15:** Booleanos y condicionales
- **Lección 16:** Condicionales y entrada del usuario
- **Lección 17:** Otras formas de entrada
- **Lección 18:** Proyecto - Tarjeta interactiva

Objetivos

- OA 1. Aplicar conceptos de Ciencias de la Computación –abstracción, organización lógica de datos, análisis de soluciones alternativas y generalización– al crear el código de una solución computacional.
- OA a. Construir y evaluar estrategias de manera colaborativa al resolver problemas no rutinarios.
- OA d. Argumentar, utilizando lenguaje simbólico y diferentes representaciones para justificar la veracidad o falsedad de una conjetura, y evaluar el alcance y los límites de los argumentos utilizados.
- OA g. Elaborar representaciones, tanto en forma manual como digital, y justificar cómo una misma información puede ser utilizada según el tipo de representación.

Referencias

- Code Studio - Code.org
- Cuantrix - Fundación Televisa
- CSTA – Computer Science Teachers Association

Lección 5: Diseñador de Minecraft

Lección en línea Ver en Code Studio

Propósito

La hora del código con Minecraft ayuda a que todos los estudiantes puedan aprender las valiosas habilidades adquiridas a través de la enseñanza de las Ciencias de la Computación. Los estudiantes participarán una hora donde experimentarán la programación en un ambiente divertido y libre de estrés, aprenderán conceptos clave de la computadora y los videojuegos.
Los estudiantes aprenderán Ciencias de la Computación a través del taller: La hora del código con Minecraft, dónde diseñarán un juego Minecraft teniendo un acercamiento a conceptos que utilizarán a lo largo del curso.

Secuencia para el aprendizaje

Ampliación del conocimiento (55 min)
Transferencia del conocimiento (5 min)

Objetivo

Los estudiantes serán capaces de:

- Entender conceptos básicos que conforman el diseño de un videojuego, de una manera divertida: estructura de código por bloques, comandos, ciclos, comandos aleatorios, eventos, agregar elementos y puntuación.

Preparación

- Computadoras con conectividad
- Realice las actividades para encontrar cualquier área potencialmente problemática para su clase.

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los profesores:

- Hora del código Minecraft

Estrategia de aprendizaje

Ampliación del conocimiento (55 min)

Se guiará a los estudiantes al curso, entrando al siguiente link - hora del código Minecraft - e identificando el curso mostrado en las imágenes a continuación:

Posterior iniciarán el curso dando clic en:

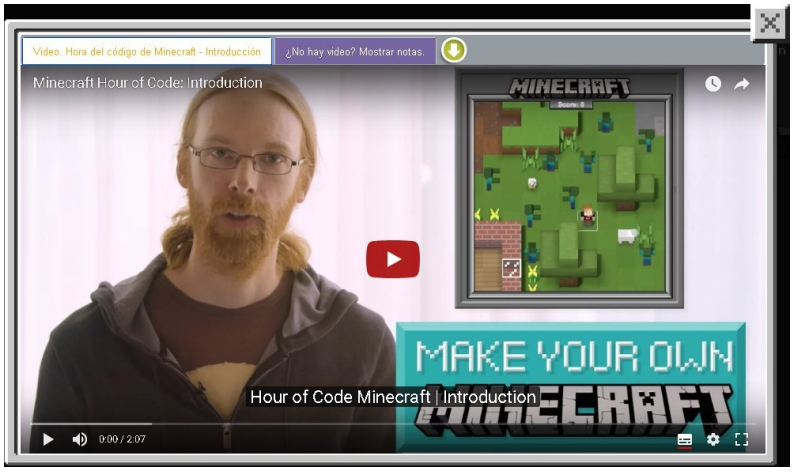
Ahora como clase, podrán explorar el taller.

Diseñador de Minecraft

Grupos: forma equipos de 2-3 personas, según la capacidad del aula.

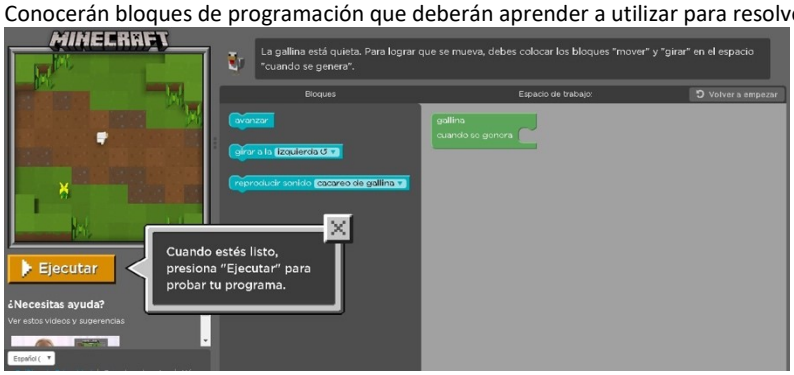
Explica: se explicará a los estudiantes sobre los elementos que conforman un videojuego, pensarán en sus videojuegos favoritos y se les preguntará ¿qué contienen? ¿qué observan? ¿cómo creen que funcionan?

Como clase observarán el video de introducción:

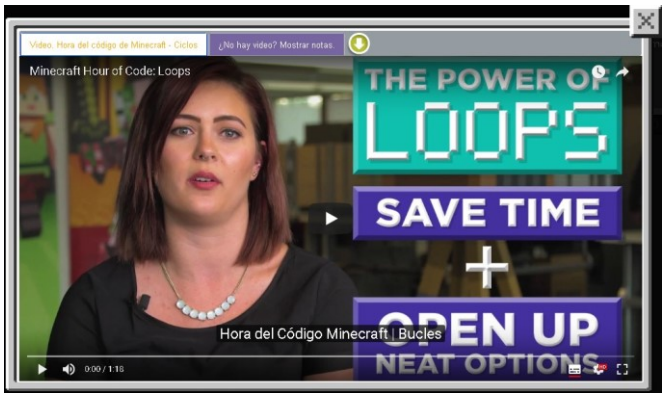


Los equipos deberán seguir avanzando en las diversas etapas y actividades del taller, además podrán seleccionar el personaje que más les guste para la Lección.

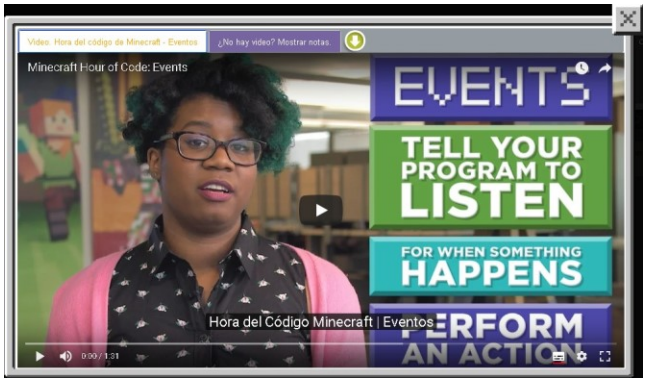
Conocerán bloques de programación que deberán aprender a utilizar para resolver diversos retos.



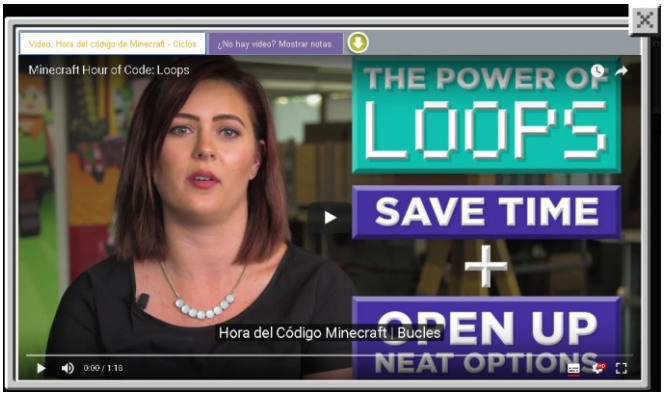
Como clase observarán el video que permitirá a los equipos aprender a utilizar ciclos de programación, que harán sus programas más eficientes, lo aprendido lo aplicarán en los ejercicios de programación.



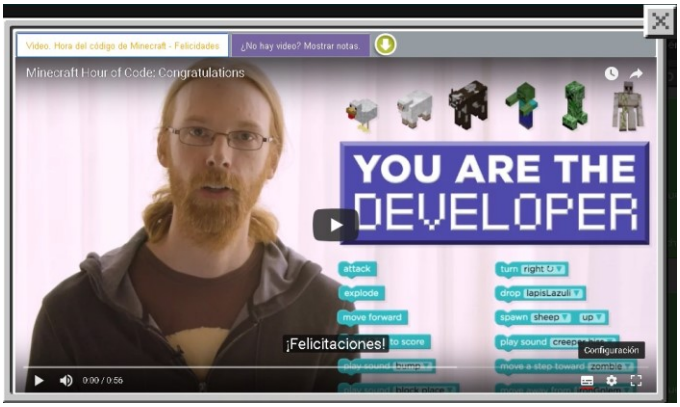
Como clase observarán el video, después los equipos harán uso de eventos que les permitirán incluir eventos dependientes de una acción en su videojuego. Lo aprendido lo aplicarán en los ejercicios de programación.



Como clase observarán el video para generar criaturas, lo aprendido lo aplicarán en los ejercicios de programación.



Como clase observarán el video de felicitaciones, en este momento habrán concluido su primer acercamiento al desarrollo de videojuegos.



Transferencia del conocimiento (5 min)

Reflexión

Pregunta a los estudiantes: ¿qué fue lo que más les gustó? y enlisten en sus cuadernos los elementos que utilizaron a lo largo del taller.
Los conceptos aprendidos los utilizarán a lo largo del curso, a mayor profundidad.
Si cuentas con más tiempo o tu clase está muy interesada en este tema pueden realizar más actividades de la hora del código con Minecraft.

Sugerencias para evaluación

Se sugiere el siguiente indicador para evaluar formativamente los aprendizajes:

- Desarrollan un programa con elementos o fases más pequeñas.
- En una conversación pueden dar un ejemplo de aplicación de elementos como: secuencias, eventos, bucles y condicionales

Lección6: Trazar formas

Lección en línea

Propósito

Los estudiantes exploran los desafíos de comunicar cómo dibujar con formas y usar una herramienta que presente cómo se aborda este problema en Game Lab. La Lección de calentamiento demuestra rápidamente los desafíos de la posición de comunicación sin algún punto de referencia compartido. En la Ampliación del conocimiento, los estudiantes exploran una herramienta de Game Lab que les permite a los estudiantes colocar formas interactivamente en la cuadrícula de 400 por 400 de Game Lab. Luego se turnan para instruir a un compañero sobre cómo dibujar una imagen oculta con esta herramienta, lo que representa muchos desafíos que los estudiantes enfrentarán al programar en Game Lab. Los estudiantes opcionalmente crean su propia imagen para comunicarse antes de una discusión de informe.
El objetivo principal de esta Lección es presentar a los estudiantes el sistema de coordenadas que usarán en Game Lab. Los estudiantes pueden tener experiencia limitada con el uso de una cuadrícula de coordenadas o pueden tener dificultades con el eje Y "invertido" en Game Lab. La herramienta de dibujo también obliga a los estudiantes a pensar en otras características de Game Lab que verán cuando comiencen a programar en la próxima Lección. Estos incluyen la necesidad de tener en cuenta el orden al dibujar, la necesidad de especificar el color y el hecho de que los círculos se colocan por su centro y cuadrados por la esquina superior izquierda. Al final de esta Lección, los estudiantes deben estar listos para transferir lo que han aprendido sobre la comunicación de posición a la programación que harán en la próxima Lección.

Secuencia para el aprendizaje

Conocimiento inicial (10 min)
Ampliación del conocimiento (35 min)
Transferencia del conocimiento (5 min)

Objetivos

Los estudiantes serán capaces de:

- Manejar las ubicaciones en la cuadrícula de coordenadas de Game Lab.
- Comunicar cómo dibujar una imagen en Game Lab, teniendo en cuenta la posición de la forma, el color y el orden.

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos

- Para los estudiantes:
- Dibujando Formas (Versión A) - Guía de actividades
- Dibujando Formas (Versión B) - Guía de Actividades

Estrategia de aprendizaje

Conocimiento inicial (10 min)

Comunicar información de dibujo



Observaciones:
Vimos muchos programas diferentes ayer, y comenzó a pensar qué tipos de programas querría crear. Cuando creamos un programa, una de las cosas que debemos hacer es dibujar todo en la pantalla. Hoy, vamos a probar eso con una computadora "estudiantil".
Pida a uno o dos voluntarios que pasen al frente del salón y actúen como "computadoras" para la Lección. Deben sentarse de espaldas para que no puedan ver lo que se proyecta. Entregue a cada voluntario una hoja de papel en blanco.
Visualización: proyecte un dibujo de muestra donde pueda ser visto por la clase.
Observaciones: Deberá explicarle a nuestra "computadora" cómo dibujar la imagen. Al final, compararemos el dibujo con la imagen real. Dé un minuto o dos a los estudiantes para describir el dibujo, mientras que los estudiantes al frente del salón tratan de dibujarlo. Después de un minuto, deténgalos y permítales que los estudiantes comparen ambas imágenes.
Pregunte: ¿cuáles son los diferentes "desafíos" o problemas que vamos a necesitar resolver para comunicar con éxito este tipo de dibujos?
Discuta:

- Los estudiantes deben escribir silenciosamente sus respuestas en sus cuadernos.
- Luego deben hablar con un compañero y finalmente con toda la clase.

Observaciones: hubo varios desafíos que necesitábamos resolver en esta Lección. Necesitamos poder comunicar claramente la posición, el color y el orden de las formas. Vamos a comenzar a explorar cómo resolver este problema.

Meta: Esta discusión tiene la intención de mencionar algunos desafíos que los estudiantes necesitarán abordar en las próximas lecciones, tales como especificar la posición, el orden y el color. Los estudiantes no necesitan decidir cómo especificarán tales cosas, pero reconocen que necesitarán un método para hacerlo.

Ampliación del conocimiento (35 min)

Dibujando con una computadora

Grupo: Agrupe a los estudiantes en parejas.
Transición: haga que un miembro de cada grupo tome una computadora y vaya a los contenidos de esta Lección. Hay un solo nivel con una herramienta de Game Lab.
Indicación: trabaje con su pareja toma dos o tres minutos para descubrir cómo funciona esta herramienta. Luego, prepárese para compartir como clase.
Discuta: después de que las parejas hayan tenido la oportunidad de trabajar con la herramienta, ejecute un reparto rápido donde los estudiantes discutan las características que notan.

Dibujando formas

Distribuya Dibujando Formas (Versión A) - Guía de actividades
a cada par, asegurándote de que un alumno (Estudiante A) reciba las dos primeras páginas y que el otro alumno (Estudiante B) reciba las dos páginas siguientes. Los estudiantes no deben mirar los papeles de los demás.

Meta: En lugar de hacer una demostración en vivo de la herramienta, use esta estrategia para que los estudiantes exploren la herramienta ellos mismos. Luego use la información para asegurarse de que todos los estudiantes conozcan las características clave de la herramienta. Los componentes más importantes son la cuadrícula y el hecho de que las coordenadas del mouse se muestran debajo del espacio de dibujo.

Configuración:
En esta Lección, los estudiantes intentarán recrear las imágenes en función de las indicaciones de un compañero. El estudiante que está ilustrando usará la herramienta de dibujo de formas en Game Lab para realizar las formas. Los estudiantes deben mantener sus dibujos ocultos el uno del otro durante toda la Lección. Al completar un dibujo, el instructor tampoco debería poder ver la pantalla de la computadora.
Dibujo 1:
Cada miembro del par debe completar su primer dibujo, turnarse para dar instrucciones y usar la herramienta. Estos dibujos no presentan formas superpuestas, pero los estudiantes pueden tener que lidiar con el hecho de que los círculos se dibujan desde la mitad y los cuadrados desde la esquina superior izquierda. Además, los estudiantes pueden simplemente luchar con la dirección del eje Y.
Discuta: da un par minutos a las parejas para discutir cualquier problema común que estén notando al tratar de completar sus dibujos.

Ubicación del origen: El origen de esta cuadrícula, así como el origen en Game Lab, se encuentra en la esquina superior izquierda. Esto refleja el hecho de que los documentos tienden a comenzar en la parte superior izquierda y aseguran que cada punto del avión tenga coordenadas positivas.

Dibujo 2:
Cada miembro del par debe describir su segundo dibujo a su compañero. Estos dibujos tienen formas superpuestas y los estudiantes tendrán que considerar el orden en que se colocan las formas y cuándo deben cambiar el color del lápiz.
Dibuje el suyo:
Si el tiempo lo permite, brinde a los estudiantes la oportunidad de crear su propio dibujo para comunicarse con su compañero.
Observaciones
Cuando hacemos imágenes, necesitamos una forma de comunicar exactamente dónde va cada forma. El plano de coordenadas nos ayuda a hacer eso. Nuestro plano de coordenadas tiene dos coordenadas, “x” y “y”. La coordenada “x” nos dice qué tan lejos está nuestra forma de la izquierda de la cuadrícula. La coordenada “y” nos dice cuán lejos está nuestra forma desde la parte superior de la cuadrícula. Los puntos negros en las formas lo ayudan a ser muy específico sobre cómo se coloca la forma en la cuadrícula.

Cuando continuar: determine si esta última Lección vale la pena en su agenda. Darles a los estudiantes la oportunidad de crear y comunicar supropio dibujo puede ayudar a reforzar su conocimiento, pero si los estudiantesobviamente están logrando los objetivos de aprendizaje de la Lección sin él, también puedes pasar al resumen para sintetizar su aprendizaje.

Transferencia del conocimiento (5 min)

Reflexión

Haga que los estudiantes reflexionen sobre cada uno de los siguientes mensajes

- ¿Qué problema está ayudando a resolver la cuadrícula en Game Lab?
- ¿Has visto formas diferentes de resolver este problema en el pasado? ¿Qué son?

Discuta: haga que las parejas compartan sus respuestas entre sí. Luego abra la discusión a toda la clase.

Observaciones
Al comienzo de la clase vimos que comunicar cómo dibujar incluso formas simples puede ser bastante desafiante. Lo que aprendimos hoy es una solución a este problema, pero hay muchas otras que podrían haber funcionado. De hecho, muchos de ustedes probablemente se dieron cuenta de que la cuadrícula en Game Lab está “volteada”. Las pantallas de las computadoras tienen diferentes formas y tamaños, al igual que el contenido que mostramos en ellas. Tenemos que acordar un punto desde el cual pueda crecer todo el contenido. Como leemos desde la esquina superior izquierda, la cuadrícula en la pantalla de una computadora también comienza en la esquina superior izquierda. También existe el beneficio de no tener que usar números negativos para hablar de ubicaciones en la pantalla. No se preocupe si esta cuadrícula volteada es un poco complicada. Tendremos mucho más tiempo para trabajar en las próximas actividades.

Lección 7: Dibujar en Game Lab

Lección en línea Ver en Code Studio

Propósito

Se les presenta a los estudiantes Game Lab, el entorno de programación y comienzan a usarlo para colocar las formas en la pantalla. Aprenden los conceptos básicos de secuenciación y depuración, así como algunos comandos simples. Al final de la Lección, los estudiantes podrán programar imágenes como las que hicieron con la herramienta de dibujo en la Lección anterior. El objetivo principal de esta Lección es dar a los estudiantes la oportunidad de acostumbrarse al entorno de programación, así como la secuencia básica y la depuración**.** Los estudiantes comienzan con una introducción al entorno de desarrollo interactivo (IDE) Gamelab, luego aprender las tres instrucciones (rect, ellipse y fill) que van a necesitar para codificar los mismos tipos de imágenes que han creado en el papel en la Lección anterior. Los niveles de desafío ofrecen una oportunidad para que los estudiantes que tienen más experiencia en programación exploren más en Game Lab.

Secuencia para el aprendizaje

Conocimiento inicial (5 min)
Ampliación del conocimiento (30 min)
Transferencia del conocimiento (10 min)

Objetivos

Los estudiantes serán capaces de:

- Usar el IDE de Game Lab para trazar diferentes formas de colores en la pantalla.
- Aprender la secuencia de código correctamente para superponer formas.
- Conocer el código de depuración escrito por otros.

Preparación

Prepara el proyector u otros medios para mostrar videos si desea verlos como una clase.
Para los profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos
- Dibujando Formas - Recursos

Para los estudiantes:

- Dibujando en el Laboratorio de Juego - Parte 1 - Video
- Dibujando en el Laboratorio de Juego - Parte 2 - Video

Vocabulario

- **Error (bug)**: Parte de un programa que no funciona correctamente.
- **Depuración (debugging)**: Encontrar y solucionar problemas en un algoritmo o programa.
- **Programa**: Algoritmo que ha sido codificado en algo que puede ser ejecutado por una máquina.

Código

- fill(color)
- ellipse(x, y, w h)
- rect(x, y, w, h)

Estrategia de aprendizaje

Conocimiento inicial (5 min)

Programación de imágenes

Observaciones

En la última Lección, creamos imágenes en la computadora organizando cuadrados y círculos en una cuadrícula. Para cada imagen que querías crear, tenías que dibujar esas imágenes manualmente, y si querías recrear una imagen era mucho trabajo. Hoy, vamos a programar la computadora para dibujar esas imágenes para nosotros. Según lo que sabe sobre las computadoras, ¿qué cree que será diferente entre decirle a una persona sobre tu imagen y contarle a la computadora tu imagen?

Dé tiempo a los estudiantes para que piensen individualmente y hablen con un compañero, luego junte la clase y escriba sus ideas en el pizarrón.

Para dar instrucciones a una computadora, necesitamos usar un lenguaje que la computadora entienda. Utilizamos HTML, que es ideal para hacer páginas web. Para hacer nuestras animaciones y juegos, usaremos una versión de Javascript que usa bloques. El entorno en el que programaremos se llama Game Lab.

Ampliación del conocimiento (30 min)

Dibujo simple en Game Lab

Grupo: Ubique a los estudiantes en parejas para programar juntos.

Transición:

Los estudiantes que son nuevos en la programación a menudo tienen algunos conceptos erróneos comunes con los que se topan. Con el fin de evitar que sigan recordando a los estudiantes sobre las siguientes cosas

- Un comando por línea
- Los comandos se ejecutan enorden de arriba abajo
- El orden de las entradas en los comandos de forma importa
- Cada comando de entrada en forma está separado por comas
- (0,0) está en la esquina superior izquierda de la pantalla
- Todos los valores x - y en la pantalla son positivos

Envía a tus estudiantes a Code Studio - dibujando en Game Lab.

Apoyo:

A medida que los estudiantes trabajan en los niveles, puede ayudarlos, pero aliéntelos a intentar dedicar algo de tiempo a resolver primero los problemas. Si necesita ayuda para apoyar a los estudiantes, consulte los ejemplos en el visor de respuestas del docente. Cuando los estudiantes alcanzan los niveles de desafío, pueden elegir perseguir uno o más de los desafíos, regresar para mejorar los niveles anteriores o ayudar a un compañero de clase.

Tour de Game Lab

Dependiendo de la edad y el nivel de comodidad de sus estudiantes, puedes optar por utilizar este nivel para recorrer el entorno como una clase completa. Asegúrese de que los estudiantes puedan encontrar las instrucciones de nivel, el área de codificación, el área de visualización y los cajones bloqueados. Esta es también una buena oportunidad para señalar algunos de los recursos útiles, como la documentación y el botón Bloquear texto.

Coloque cuadrados en las esquinas

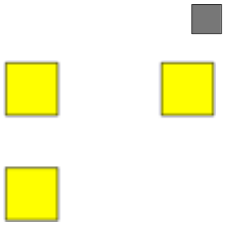
Una gran parte de usar Game Lab es entender la posición. Recuerda que siempre puede activar la cuadrícula o desplazarse con el mouse para ayudar a encontrar laposición x-y que desea.

Color de relleno

También puede hacer que los rectángulos tengan diferentes colores con fill (rellenar). Establecerá el color para cada forma que viene después en el código.

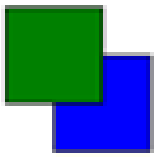
Hacer esto

- Mire el código que establece el color.
- Cambie el color de azul a amarillo.
- Agregue un nuevo cuadrado arrastrando un bloque rect al área de código debajo del comando de relleno. El nuevo cuadrado puede ir a cualquier ubicación en la pantalla que desees.



El orden importa

En Game Lab, importa en qué orden se encuentre el código. Se dibujan nuevas formas encima de las anteriores, cubriendo las formas que se dibujan primero. Puedes ver la diferencia cuando usa más de un color en tu código

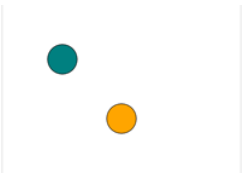


Elipse

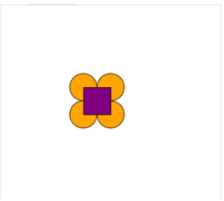
Puedes usar ellipse para hacer un círculo.

Depuración

A menudo, el código no funciona la primera vez que se ejecuta y los programadores tienen que depurarlo. Se supone que el siguiente código hace que la imagen aparezca a la derecha, pero el programador se confundió sobre en qué orden debería estar el código, y cómo colocar un cuadrado en el lugar correcto de la pantalla.



```
1. rect(175, 175);
2. fill("orange");
3. ellipse(150,150);
4. ellipse(200,150);
5. ellipse(150,200);
6. ellipse(200,200);
7. fill("purple");
```



Formas simplificadas

Los comandos rect y ellipse que se presentaron a los estudiantes en estas actividades son versiones simplificadas de los comandos completos que verán más adelante. Esto permite que la clase se centre únicamente en la ubicación de las formas en el plano de coordenadas antes de preocuparse también por el ancho y la altura de esas formas.

Transferencia del conocimiento (10 min)

Compartir dibujos

Objetivo: Los estudiantes pueden ver la variedad de cosas diferentes que pueden crear con simples dibujos de formas.
Compartir: Una vez que los estudiantes hayan completado sus dibujos, pide que compartan sus dibujos con la clase.

Ticket de salida

Meta: Los estudiantes comparten los trucos que aprendieron a medida que pasaron por niveles.
Indicación: Hoy aprendieron a dibujar en Game Lab por primera vez. ¿Qué tipo de consejo compartirán con sus amigos que iban a aprender a dibujar en Game Lab para que sea más fácil para ellos? Escríbanlo en un pedazo de papel.

Recopilar: Recopila las respuestas de los estudiantes y selecciona algunas que puedan ser útiles para que los escuchen todos los estudiantes. Comparte esos al comienzo de la próxima clase.

Este es un buen acertijo para usar como una evaluación de si los estudiantes entienden o no los fundamentos dela secuencia y el posicionamiento de formas en Game Lab.

Lección8: Formas y aleatorio

Lección en línea Ver en Code Studio

Propósito

En esta Lección, los estudiantes continúan desarrollando su familiaridad con Game Lab al manipular el ancho y la altura de las formas que usan para dibujar. La Lección comienza con una discusión que conecta la funcionalidad de bloque expandido (por ejemplo, formas de diferentes tamaños) con la necesidad de más entradas de bloque, o "parámetros". Los estudiantes aprenden a dibujar con versiones de ellipse() y rect() que incluyen parámetros de ancho y alto. También aprenden a usar el bloque background(). Al final del progreso, se les presenta a los estudiantes el bloque randomNumber(). Combinando todas estas habilidades, los estudiantes dibujarán una serpiente arcoiris al azar al final de la Lección.
Esta Lección les da a los estudiantes la oportunidad de expandir ligeramente sus habilidades de dibujo mientras continúan desarrollando habilidades de programación de propósito general. Tendrán que razonar sobre el plano de coordenadas x-y, considerar el orden de su código y aumentar ligeramente la complejidad de sus programas. El bloque randomNumber() es importante para la próxima clase, donde los estudiantes aprenden a almacenar valores utilizando variables. Esta Lección debe enfocarse principalmente en la construcción de habilidades. Si los estudiantes pueden completar la serpiente arco iris de forma independiente, entonces tienen las habilidades que necesitarán para las próximas actividades.

Secuencia para el aprendizaje

Conocimiento inicial (5 min)
Ampliación del conocimiento (40 min)
Transferencia del conocimiento (5 min)

Objetivos

Los estudiantes serán capaces de:

- Usar y razonar sobre el dibujo de comandos con múltiples parámetros.
- Generar y usar números aleatorios en un programa.

Preparación

- Revisa la secuencia de niveles en Code Studio.

Para los profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos
- Formas y parámetros - Recurso
- Formas y parámetros - Recurso

Vocabulario

- **Parámetro:** Dato que se considera como imprescindible y orientativo para lograr evaluar o valorar una determinada situación.

Código

- background(color)
- ellipse(x, y, w, h)
- rect(x, y, w, h)
- randomNumber()

Estrategia de aprendizaje

Conocimiento inicial (5 min)

1. ellipse()
2. randomNumber(5, 10)
3. rect()

Formas de diferentes tamaños

Indicación:

Nuestros bloques ellipse y rect cada uno tienen dos entradas que controlan dónde están dibujadas: la posición x-y. Si quisiera que estos comandos dibujarán una variedad más amplia de rectángulos y elipses, ¿qué entradas adicionales podría necesitar para proporcionar estos bloques? ¿Qué controlaría cada entrada adicional?

Discute:

Los estudiantes deben intercambiar ideas en silencio, luego compartir con un compañero, después compartir con toda la clase. Registra ideas mientras los estudiantes las comparten en el pizarrón.

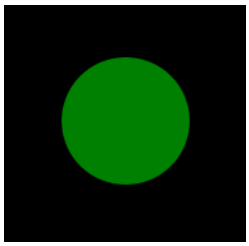
Observaciones

Esta fue una buena lista de ideas. Si queremos que nuestros bloques dibujen formas de diferentes maneras necesitarán más insumos que nos permitan decirles cómo dibujar. Las entradas o aperturas en nuestros bloques tienen un nombre formal, parámetros, y hoy vamos a aprender más sobre cómo usarlas.

Meta: Esta discusión introduce la palabra de vocabulario "parámetro" y también ayuda a motivar su uso. Los estudiantes verán que las versiones del bloque ellipse() y rect() en esta Lección tienen parámetros adicionales, así como el bloque randomNumber() que tiene dos parámetros. Los estudiantes pueden decir que quieren insumos para el tamaño de las formas, su color, etc. Durante esta conversación, vincule las conductas que los estudiantes desean con las entradas que el bloque necesitaría. Por ejemplo, si desea que los círculos tengan un tamaño diferente, el bloque necesitará una entrada que le permita al programador decidir qué tan grande debe ser.

Ampliación del conocimiento (40 min)

Programación de imágenes



Transición:

Mover a los estudiantes a Code Studio y realizar las actividades:

- Resumen de la Lección
- Formas y parámetros
- Números al azar

Compartir:

Si algunos estudiantes se toman más tiempo para trabajar en sus proyectos, dales la oportunidad de compartir sus serpientes arcoíris más complejas. Enfoque la conversación sobre qué parámetros están manipulando los estudiantes o aleatorizando para crear sus dibujos.

Fondo

A veces querrá llenar toda la pantalla con un color. Para eso, puedes usar el bloque background. Cubra todo en la pantalla con el color que elijas.

El bloque de fondo se dibujará encima de todo lo que ya está en tu dibujo, por lo que el orden en el código es importante.

Números al azar

randomNumber() elija un número aleatorio entre un valor mínimo y máximo. Puede usar este bloque en lugar de escribir en el número específico. Si realiza sus dibujos con números aleatorios, se verá un poco diferente cada vez que ejecute el programa.

Transferencia del conocimiento (5 min)

Reflexión

Preguntar:

Pida a los estudiantes que reflexionen sobre el desarrollo de las cinco prácticas de Descubrimientos CS (Resolución de problemas, Persistencia, Creatividad, Colaboración, Comunicación). Elija una de las siguientes indicaciones según lo considere apropiado.

- Elige una de las cinco prácticas en las que crees que demostraste crecimiento en esta Lección. Escribe algo que hiciste que ejemplifica esta práctica.
- Elige una práctica en la que pienses que puede seguir creciendo. ¿Qué te gustaría mejorar?
- Elige una práctica que pensaste que era especialmente importante para la Lección que completamos hoy. ¿Qué lo hizo tan importante?

Lección 9: Variables

Lección en línea Ver en Code Studio

Propósito

En esta Lección, los estudiantes aprenden a usar variables para etiquetar un número en su programa o guardar un valor generado aleatoriamente. Los estudiantes comienzan la lección con una descripción muy básica del propósito de una variable. Luego, los estudiantes completan un progreso de nivel que refuerza el modelo de una variable como una forma de etiquetar o nombrar un número. Los estudiantes usan variables para guardar un número aleatorio para ver que las variables realmente almacenan o guardan sus valores, lo que les permite usar el mismo número aleatorio varias veces en sus programas. En esta lección, los estudiantes verán por primera vez variables, y no se espera que entiendan completamente cómo funcionan para cuando ésta termine. Por lo tanto, los estudiantes deben terminar esta lección sabiendo que las variables son una forma de etiquetar un valor en sus programas para que puedan ser reutilizados o referenciados más adelante. En la siguiente lección, los estudiantes conocerán los sprites, los que deben ser referenciados por una variable.

Usar variables para manipular dibujos es una habilidad sorprendentemente desafiante que requiere una gran cantidad de previsión y planificación. Si bien los estudiantes usarán o modificarán muchos programas en esta lección, no se espera que creen programas que usan variables para modificar las características de un dibujo. En lecciones posteriores, los estudiantes ampliarán su comprensión de las variables y formas más avanzadas.

Secuencia para el aprendizaje

Conocimiento inicial (10 min)
Ampliación del conocimiento (30 min)
Transferencia del conocimiento (5 min)

Objetivos

Los estudiantes serán capaces de:

- Identificar una variable como una forma de etiquetar y referenciar un valor en un programa.
- Usar variables en un programa para almacenar una información que se usa varias veces.
- Identificar el motivo y solución de errores comunes encontrados al programar con variables

Preparación

- Revisa el progreso de nivel en Code Studio.

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos
- Nombrar variables - Recurso
- Variables - Recurso

Para los estudiantes:

- Introducción a las variables – Video

Vocabulario

- **Variable:** Un marcador de posición para una información que puede cambiar.

Código

- Declara variable

Estrategia de aprendizaje

Conocimiento inicial (10 min)

Etiquetas y valores

Video:
Como clase, mire el video que presenta las variables.
Revisión:
Los estudiantes pueden revisar los puntos más importantes del video en el nivel 4. Los estudiantes no necesitan comprender todas estas ideas ahora mismo, pero pueden usar este nivel como referencia durante toda la Lección.
Observaciones
Hay mucho que aprender sobre las variables. Hoy vamos a usarlas para ayudarnos a hacer dibujos. Sin embargo, lo más importante es simplemente ver cómo dar a una etiqueta un valor nos ayuda a escribir programas.

Evitar la carga frontal: si bien esta Lección comienza con dos recursos que explican cómo funcionan las variables, es probable que sean más significativas para los estudiantes una vez que hayan utilizado variables en los programas de dibujo. Asegúrese de que los estudiantes sepan que estos recursos están disponibles, y luego revíselos -si lo desea- al final de la Lección para ayudarlos a crear sentido.

Ampliación del conocimiento (30 min)

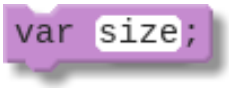
Programación con variables

Dirige a los estudiantes a los Code Studio y aborden las siguientes actividades:

- Introducción a las variables
- Variables
- Nombrar variables
- Desafío: dibujar una imagen
- Desafío: actualización de variables

Variables

Una variable te permite almacenar un solo valor en la memoria de su computadora con un nombre descriptivo. El uso de variables le permite consultar fácilmente el mismo valor muchas veces en su programa o guardar un número al que le gustaría consultar más adelante.

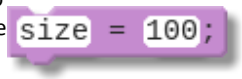


Creación de variables

El comando var creará una nueva variable con la etiqueta que le da. Esta variable tiene la etiqueta size.

Asignación de valores

El operador de asignación = asignará un nuevo valor a su variable. Este comando asignó 100 al tamaño variable. La variable siempre debe estar en el lado izquierdo. Usted leería este comando como “obtiene tamaño 100”, ya que el tamaño obtiene un nuevo valor de 100. Cualquier valor antiguo que podría haber sido asignado se pierde para siempre. A las variables también se les puede asignar un número aleatorio. Esto le permite guardar un único valor aleatorio para que pueda usarlo tantas veces como desee en su programa.



Transferencia del conocimiento (5 min)

Reflexión

Indicación:

Dé a los estudiantes las siguientes instrucciones

- ¿Cuál es tu propia definición de variable?
- ¿Por qué las variables son útiles en los programas?

Discute:

Haga que los estudiantes escriban sus ideas de forma independiente, posteriormente, invítalos a compartirlas con los compañeros del grupo.

En sus cuadernos:

¿qué conexiones observas entre las variables y el modelo entrada - almacenamiento - procesamiento - salida de una computadora?

Meta: Utilizar esta discusión para evaluarlos modelos mentales de una variable de los estudiantes. Es posible que desee que los estudiantes escriban susrespuestas para que pueda recopilarlasy revisarlas más tarde. Debería verprincipalmente que entienden que las variables pueden etiquetar o nombrar un número para que pueda ser utilizadomás adelante en sus programas. Si bienhay otras propiedades de una variable que los estudiantes pueden haber aprendido, esta es la más importante antes de pasar a la siguiente Lección.

Sugerencias de evaluación

Se sugiere el siguiente indicador para evaluar formativamente los aprendizajes:

- Nombran las variables en relación con su significado (nemotécnico).
- Utilizan las variables para hacer operaciones.

Lección 10: Sprites

Lección en línea Ver en Code Studio

Propósito

Para crear imágenes más interesantes y detalladas, se les presenta a los estudiantes el objeto sprite. A cada sprite se le puede asignar una imagen para mostrar, y los sprites también hacen un seguimiento de múltiples valores acerca de ellos mismos, lo que resultará útil en el camino al hacer animaciones. Hacer un seguimiento de muchas formas y las diferentes variables que controlan aspectos de esas formas puede ser muy complejo. Habrá muchas variables con diferentes nombres. En cambio, los científicos informáticos crearon algo llamado objeto que permite un nombre de variable para controlar tanto la forma como todos sus aspectos. En Game Lab usamos un cierto tipo de objeto llamado sprite. Un sprite es solo un rectángulo con propiedades para controlar su aspecto. Las propiedades son las variables que están unidas a un sprite. Puede acceder a ellos mediante notación de puntos.
Usando la pestaña Animación, los estudiantes pueden crear o importar imágenes para ser usadas con sus sprites. Más adelante, estos sprites se convertirán en una herramienta útil para crear animaciones, ya que sus propiedades se pueden cambiar y actualizar a lo largo del curso de un programa.

Secuencia para el aprendizaje

Conocimiento inicial (5 min)
Ampliación del conocimiento
Transferencia del conocimiento (5-10 min)
Evaluación

Objetivos

Los estudiantes serán capaces de:

- Asignar un sprite a una variable.
- Usar notación de puntos para actualizar las propiedades de un Sprite.
- Crear una escena estática combinando sprites, formas y texto.

Preparación

(Opcional) Imprima una copia de la guía de actividades para cada estudiante

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los profesores:

- Pestaña de Animación – Recurso
- CSD Unidad 3 - Animaciones y Juegos Interactivos
- Sprites – Recurso

Para los estudiantes:

- Introduction to Sprites – Video
- La pestaña de Animación – Video

Vocabulario

- **Propiedad**: Atributos que describen las características de un objeto.
- **Sprite:** Un personaje gráfico en la pantalla con propiedades que describen su ubicación, movimiento y apariencia.

Código

- drawSprites(group)
- var sprite = createSprite(x, y, width, height)
- sprite.scale

Estrategia de aprendizaje

Conocimiento inicial (5 min)

¿Cuánta información?

Piense, empareje, comparta:

Hasta ahora solo hemos escrito programas que ponen formas simples en la pantalla. Haz una lista de todas las diferentes piezas de información que has utilizado para controlar cómo se dibujan estas formas.

Preguntar:

¿Qué pasaría si quisiéramos crear programas con imágenes más detalladas, tal vez incluso con personajes con los que pudiéramos interactuar? ¿Qué otras piezas de información pueden necesitar en su código?

Observaciones

Hoy aprenderemos cómo crear personajes en nuestras animaciones llamados sprites. Estos sprites se almacenarán en variables, al igual que almacenaron números en el pasado, pero los sprites pueden contener muchos datos, lo que le permitirá crear programas mucho más interesantes (¡y eventualmente animados!).

El objetivo aquí es hacer que los estudiantes piensen en todos los diferentes valores que forman una única forma en la pantalla, y cuántos valores más pueden necesitar para controlar un carácter más detallado en un programa. Si los estudiantes están luchando para encontrar ideas, puede usar algunas de las siguientes indicaciones: *¿Cómo le dice a una figura a dónde ir a la pantalla? ¿Cómo le dices a una forma qué tamaño necesita ser? ¿Cómo le dices a una forma de qué color debería ser? ¿Qué hay de su esquema?* ¿Qué sucede si quiere cambiar alguno de esos valores durante su programa o controlar otros elementos, como la rotación?

Ampliación del conocimiento

Introducción a Sprites

Distribuir:

Sprite planificación de escena - Guía de actividades. Los estudiantes pueden usar esta hoja para planear la escena de Sprite que crean al final de esta Lección, pero la planificación también se puede completar en papel borrador.

Transición:

Enviar estudiantes a Code Studio y que realicen los Niveles de Code Studio. Sprite.

- Creando Sprites
- La pestaña de animación
- Escenas de sprite
- Extiende tu escena

El sprite es un tipo de datos llamado **objeto**. Si bien todavía no estamos introduciendo explícitamente el concepto de objetos, los estudiantes deben comprender que un sprite es un tipo de valor diferente de los que hemos visto anteriormente, uno que puede contener referencias a muchos más valores. Para los estudiantes que sienten curiosidad sobre si hay otros objetos en nuestros programas, pídales que vean si hay más bloques en la cajade herramientas que siguen la misma notación de *puntos* (como World.widthy World.height)

Sprites

Crear Sprites:

El bloque 

crea un nuevo sprite y lo asigna a una variable. El nombre predeterminado es sprite, por lo que querrá cambiarlo a algo más significativo.

Dibujo de Sprites:

Los Sprites solo aparecen en la pantalla cuando los dibujas `drawSprites()` allí. Llamar al comando dibuja todos los sprites creados en la pantalla.

Imágenes:

Una vez que haya creado un sprite, puedes usar el sprite. setAnimation() comando para cambiar el aspecto de su sprite de un rectángulo a una imagen. Todas las imágenes que ha cargado en la pestaña Animación aparecen en el sprite.setAnimation() menú desplegable.

Suba su propia imagen:

También puedes usar la pestaña Animación para cargar o dibujar su propia imagen.

- Haga clic y luego cargar una imagen.
- Selecciona el archivo de tu computadora.
- Cambie el nombre de su imagen para que sea fácil de recordar. Para cambiarle el nombre, haga clic en el texto debajo de la imagen.

- De vuelta en el modo de código, agregue un bloque sprite.setAnimation() para que su sprite use su nueva animación.

Cambiar el tamaño con escala

En el cajón de Sprites de la caja de herramientas, verá un nuevo bloque llamado sprite.scale. Este comando le permite cambiar el tamaño de un sprite en relación con su tamaño original. sprite.scale = 1 es el tamaño normal. sprite.scale = 0.5 hace que su sprite sea la mitad de grande, mientras sprite.scale = 2 lo hace dos veces más grande.

Agregar texto

El bloque text le permite colocar texto en cualquier lugar que desee en la pantalla. Cambie este texto en el bloque provisto a otra cosa y agregue un segundo text bloque para escribir en una parte diferente de la pantalla.

Consejo:

El tamaño de texto predeterminado es bastante pequeño, pero puede usar el bloque textSize para cambiar eso. También puede usar el bloque fill para cambiar el color de tu texto.

Transferencia del conocimiento (5-10 min)

Compartir

Permite a los estudiantes compartir sus escenas de Sprite. Alienta a los estudiantes a reflexionar sobre sus escenas e identificar las formas en que les gustaría mejorar.

Evaluación

Evaluar escenas de sprite

Para evaluar las escenas de Sprite, pida a los estudiantes que hablen de su código. Verifique para asegurarse que los estudiantes sepan por qué secuenciaron su código de la manera en que lo hicieron y, en particular, busquen “código muerto” o código que no afecte a la escena final. En este punto, es probable que los estudiantes sigan dibujando formas antes de dibujar el fondo (que luego no se verá) o que estén llamando drawSprites() varias veces (solo se debe llamar una vez).

Lección 11: Ciclo de dibujo aleatorio

Lección en línea Ver en Code Studio

Propósito

En esta Lección, se presenta a los estudiantes el ciclo de dibujo, uno de los paradigmas de programación principales en Game Lab. Para comenzar la lección, los estudiantes miran algunos flipbooks físicos para ver que tener muchos marcos con diferentes imágenes crea la impresión de movimiento. Luego, los estudiantes ven un video que explica cómo el ciclo de dibujo en Game Lab ayuda a crear la misma impresión en sus programas. Los estudiantes combinan el ciclo de dibujo con números aleatorios para manipular algunas animaciones simples con puntos y luego con sprites. Al final de la Lección, los estudiantes usan lo que aprendieron para actualizar su escena de sprites de la lección anterior.
El ciclo de dibujo es un componente central de Game Lab. El hecho de que el entorno de Game Lab llama repetidamente a esta función muchas veces por segundo (por defecto 30) es lo que permite a la herramienta crear animaciones. Esta lección tiene dos objetivos, por lo tanto. El primero es que los estudiantes vean cómo la animación en general depende de mostrar muchas imágenes ligeramente diferentes en una secuencia. Para ayudar a los estudiantes a tener flipbooks físicos que pueden usar, un video. El segundo objetivo es que los estudiantes comprendan cómo el ciclo de dibujo les permite crear este comportamiento en Game Lab. Los estudiantes deben dejar la Lección entendiendo que los comandos del ciclo de dibujo se invocan después de todos los demás códigos, pero luego se los llama repetidamente a una velocidad de fotogramas. Los estudiantes tendrán la oportunidad de continuar desarrollando una comprensión de este comportamiento en las próximas dos lecciones.

Secuencia para el aprendizaje

Conocimiento inicial (5 min)
Ampliación del conocimiento (60 min)
Transferencia del conocimiento (5 min)

Objetivos

Los estudiantes serán capaces de:

- Explicar qué es una animación y cómo crea la ilusión de un movimiento simple.
- Explicar cómo el ciclo de dibujo permite la creación de animaciones en Game Lab.
- Utilizar el ciclo de dibujo en combinación con el comando randomNumber (), las formas y los sprites para realizar animaciones simples.

Preparación

- Imprima y prepare los manipulativos
- Prepare el video

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos
- Ejemplo de Flipbook - por Marnic Bos - Video
- Flipbook de Puntos Aleatorios
- Flipbook de Sprite Aleatorio
- El ciclo de dibujo – Recurso

Para los estudiantes:

- Introducción al ciclo de dibujo – Video

Vocabulario

- **Animación:** Una serie de imágenes que crean la ilusión de movimiento al mostrarse rápidamente una tras otra.
- **Marco:** Una sola imagen dentro de una animación.
- **Frecuencia de fotogramas**: La velocidad a la que se muestran los fotogramas de una animación, normalmente medida en fotogramas por segundo.

Código

- function draw() { }
- World.frameRate

Estrategia de aprendizaje

Conocimiento inicial (5 min)

Video:
Mostrar ejemplo de Flipbook - Video
Indicación:
Este video muestra un flipbook para hacer animaciones. Con tus propias palabras, explica: ¿cómo está funcionando? ¿Por qué "engaña a nuestros ojos" para que se piense que algo se está moviendo?
Discuta:
Haga que los estudiantes escriban sus ideas de forma independiente, luego compártanlo con sus compañeros, luego como un grupo completo.
Observaciones:
Vamos a empezar a aprender cómo hacer animaciones, no solo imágenes fijas. Para hacer esto, necesitamos una forma que nuestros programas dibujen muchas imágenes por segundo. Para hacer esto, tendremos que aprender una nueva herramienta importante.

Objetivo: Esta discusión debe presentar algunas ideas clave sobre la animación. Los estudiantes deben entender que laclave es ver muchas imágenes seguidas que son ligeramente diferentes. Introduzca la palabra de vocabulario "marco" como una de esas imágenes. Luego haga la transición al hecho de que pronto los estudiantes crearán sus propias animaciones.

Ampliación del conocimiento (45 min)

Video:
Mira el video de introducción a ciclos de dibujo,
Dirige a los estudiantes a CodeStudio y realiza los Niveles de Code Studio.

- Formas y el lazo
- Sprites y Draw Loop
- Sprite Propiedades
- Anima tu escena

Transferencia del conocimiento (10 min)

Compartir:
Los estudiantes sólo necesitan hacer pequeños cambios en sus proyectos (p. Ej., Agitando un solo sprite), pero pide que compartan con un compañero o como clase completa.
Preguntar:
Haga que los estudiantes respondan a las siguientes instrucciones:

- ¿Qué es una animación?
- ¿Por qué el ciclo de dibujo nos ayuda a hacer animaciones?
- ¿Cuáles son algunos errores o errores comunes que debemos tener en cuenta a medida que seguimos programando con el ciclo de dibujo?

Revisión:
Regrese a los recursos que los estudiantes vieron al comienzo de la lección y aborda los conceptos erróneos que han surgido en la lección.

Meta: Use estos avisos para evaluar si los estudiantes han entendido los principales objetivos de aprendizaje de la Lección.
Conceptos clave: Existen muchos conceptos erróneos comunes con el ciclo de extracción. Asegúrese de que los estudiantes entiendan lo siguiente

- El ciclo de extracción se ejecutadespués de todos los demás códigos en su programa. En realidad, no importa donde se encuentre en su programa
- The Game Draw es ejecutado porGame Lab a una tasa de cuadrosconstante de 30 fotogramas por segundo. En realidad, no necesitallamar a la función usted mismo
- Los "marcos" en Game Lab se pueden considerar hojas de transparencia, A menos que dibujesun fondo, todas tus nuevas formaso sprites simplemente aparecerán encima de tus antiguos
- Solo debe de tener un ciclo deextracción en su programa

Sugerencias para la evaluación

Se sugiere el siguiente indicador para evaluar formativamente los aprendizajes:

- Ejecutan ciclos (loops), a partir de un patrón que se repite en una secuencia.
- Evalúan su trabajo de manera colaborativa.
- Utilizan procesos de depuración
- Aprovechan recursos disponibles en las áreas de trabajo.

Lección 12: Patrón de contador desconectado

Lección sin conexión

Propósito

Los estudiantes exploran el comportamiento subyacente de las variables a través de una lección desconectada. Usando tarjetas de notas y cadenas para simular variables dentro de un programa, los estudiantes implementan algunos programas cortos. Una vez que se sienten cómodos con esta sintaxis, los estudiantes usan el mismo proceso con las propiedades de los sprites, haciendo un seguimiento del progreso de un sprite en la pantalla.

El razonamiento sobre las variables puede ser complicado, especialmente para los nuevos programadores. En esta Lección, los estudiantes completan una lección desconectada utilizando manipulativos físicos (tarjetas y cuerdas) para construir un modelo mental de cómo la información puede almacenarse en una variable y ser manipulada por un programa. Este modelo luego se extiende a las propiedades de los sprites, que mantienen los valores de una manera similar. Esta Lección presenta la sintaxis y los conceptos que los estudiantes podrán "conectar" en la siguiente lección.

Secuencia para el aprendizaje

Conocimiento inicial (15 min)
Ampliación del conocimiento (20 min)
Ampliación del conocimiento 2 (30 min)
Transferencia del conocimiento (10 min)

Objetivos

Los estudiantes serán capaces de:

- Describir la conexión entre la actualización de las propiedades de ubicación de un sprite y el movimiento de los sprites en la pantalla.
- Leer y seguir los pasos de un breve programa escrito en pseudocódigo que manipula los valores de las variables.

Preparación

- Preparar materiales para etiquetas y valores: Fichas, post-its o trozos de papel, etc.
- Preparar materiales para conectores: piezas de cuerda, plumas o limpiapipas, etc.
- Tablero de variables desconectado - Manipulador para cada grupo o reúne papel para que los estudiantes lo utilicen para hacer sus tableros.
- Revisar las reglas de la Lección Desconectada de Variables para asegurarte de que las comprenda y prepárate para responder preguntas, especialmente si las demostrarás tú mismo.
- Sacar copias de Variables desconectada- Guía de actividades para cada alumno.

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los estudiantes:

- Variables desconectada- Guía de actividades

Vocabulario

- **Expresión:** Cualquier unidad de código válida que resulta en un valor.
- **Variable:** Un marcador de posición para una información que puede cambiar.

Estrategia de aprendizaje

Conocimiento inicial (15 min)

Pregunta:
En la última Lección, usamos el ciclo de dibujo para hacer que nuestros sprites se muevan en la pantalla. ¿Cuáles son otras formas en que podríamos hacer que nuestros sprites se muevan?
Ponga las ideas de los estudiantes en el pizarrón.
Observaciones
En las próximas actividades, vamos a ver muchas formas de mover nuestros sprites. Para hacer eso, necesitamos aprender un poco más sobre las variables y cómo funcionan. Hoy vamos a hacer una Lección con variables y propiedades de sprites que nos ayudarán a hacer estos tipos de movimiento. A medida que avanzamos en la lección, piensen cómo lo que está aprendiendo puede ayudarlo a hacer que sus sprites se muevan de la manera que desee.

Esta Lección está diseñada para abordar muchos **conceptos erróneos comunes** con variables y memoria.
Conceptos erróneos comunes

- Las variables pueden tener valores múltiples (no pueden, las variables tienen como máximo un valor)
- Las variables "recuerdan" los valores antiguos (no, por lo tanto, los valores antiguos se eliminan enla Papelera)
- Las variables se conectan después de un comando como "x=y" (Esto puede surgir más tarde en el cursocuando los estudiantes usan sprites y se abordarán en gran detalle.
    - Por ahora, los estudiantes se ven obligados a crear nuevas tarjetas de valor cada vez porque incluso para una declaración como x=y no hay "conexión" entre las variables formadas)
- Las variables tienen expresiones (por ejemplo, 1+5). Las variables solo contienen valores, las expresiones se calculan de antemano. Es por eso que las tarjetas de valores solo se crean una vez que los estudiantes tienen un solo valor. Aplicar esta regla de cerca

Ampliación del conocimiento (20 min)

Variables Lección desconectada
Grupo:
Agrupar a los estudiantes en parejas.
Distribuir:
Entregar a cada par:

- Un conjunto de etiquetas / valores y conectores
- Una sola hoja de papel para crear su pizarrón
- 2 copias de las Variables desconectada - Guía de actividades.

Pantalla:
Muestre las reglas de la página principal de la guía de actividades y escribe los primeros dos programas de la segunda página.
Observaciones:
Hoy vamos a trabajar en un mundo de etiquetas, valores y conectores entre ellos. Para simular este mundo, usarás los trozos de papel y cuerda que les he dado. Para empezar, tendremos que configurar nuestros tableros, y luego repasaremos cómo funcionan los comandos para este mundo.
Demuestra:
Muestre a la clase cómo dividir sus tablas en 3 secciones y etiquételas en secuencia, como se muestra en la primera página de la guía de actividades.
Apoyo:
Los estudiantes deben trabajar en los dos primeros programas como un grupo, mientras hace referencia a los pasos en la guía de actividades. A medida que los equipos se sienten cada vez más cómodos al completar los comandos, alientan a los equipos a ejecutar cada comando de forma independiente antes de comparar sus tableros con un par compañero. El objetivo es solo asegurarnos de que todos tengan la oportunidad de comprender los pasos de la Lección.
Observaciones:
Ahora es su turno de intentar ejecutar algunos de estos programas por tu cuenta. En la parte inferior de la página hay dos programas más que puede ejecutar. Para cada uno, debe ejecutar el programa para averiguar el estado final del programa. En otras palabras, debe saber qué etiquetas están conectadas a qué valores. Una vez que llegue al final de cada programa, puedes comparar tus resultados con un compañero. Si no estás de acuerdo, vuelve para ver si puedes encontrar dónde perdió la pista.
Apoyo:
Los estudiantes deben trabajar en parejas a través de los dos programas en la guía de actividades. Haga un chequeo para asegurarse de que todos estén de acuerdo con el estado final del programa (qué etiquetas están conectadas con qué valores). Si los estudiantes no están de acuerdo, refuerza la necesidad de depurar al leer el código retrocediendo y rastreando cada paso.

Ampliación del conocimiento 2 (30 min)

Sprite propiedades

Distribuir:
Propiedades de Sprite en Variables desconectada- Guía de actividades si no lo ha incluido en la guía de actividades original.
Demuestre:
Muestra a los estudiantes las reglas de lección en la guía de actividades. Trabaje en el programa 5 como grupo y muestre cómo crear una nueva tarjeta Sprite y conectarla a tu tarjeta de etiqueta variable y a las tarjetas de propiedad de sprite. Asegúrate de que los estudiantes comprendan que deberían crear una nueva tarjeta sprite cada vez que vean el comando createSprite y que dibujen sus sprites en la cuadrícula cada vez que vean el comando drawSprites.

Objetivo: Los estudiantes deben ver que los comandos, como x=x+1 mover un sprite de una manera deliberadaa través de la pantalla, en posición al movimiento aleatorio que vieron en laLección anterior

Apoye:
Los estudiantes deben trabajar en los últimos dos programas usando sus materiales manipulables.
Pregunta:
¿Cómo se movió el sprite a través de la red en el Programa 3? ¿Cómo se movió el sprite a través de la red en el Programa 4 de la guía de actividades?
Después de que los estudiantes hayan completado las preguntas de reflexión, deben comparar con otro par, luego discutirlo como clase.

Transferencia del conocimiento (10 min)

Discusión

Indicación:
Hoy vimos algunas pistas de cómo podríamos programar los tipos de movimiento que queremos para nuestros sprites. ¿Cuáles son algunos problemas que aún tenemos que resolver para que el sprite parezca moverse en la forma que deseo?
Permita que los estudiantes hagan una lluvia de ideas sobre problemas y anótelos en el pizarrón.
Preguntar:
Elija uno o dos de estos problemas y comience a pensar en algunas formas en que podría ser resuelto.
Dé tiempo a los estudiantes para que hagan una lluvia de ideas individualmente antes de compartir sus soluciones.
Observaciones:
Estas son buenas ideas. En la próxima lección, vamos a ver cómo podemos usar algunas de las cosas que hemos aprendido hoy para hacer que nuestros sprites se muevan de muchas maneras diferentes.

El objetivo de la discusión no es hacer que los estudiantes piensen en soluciones a todos los problemas, sino que los identifiquen, preparándolospara la Lección del patrón contrario. Algunos programas que los estudiantespuedan ver son que las caras sonrientespermanecieron en la pantalla, en lugar de moverse, que solo podían trasladarse la izquierda y la derecha o hacia arriba o hacia abajo, o que las imágenes se detienen después de un cierto periodo de tiempo.

Sugerencias para evaluación

Se sugiere el siguiente indicador para evaluar formativamente los aprendizajes:

- Describen de manera explícita, cada paso que compone un proceso secuencial.

Lección 13: Movimiento de sprite

Lección en línea Ver en Code Studio

Propósito

Al combinar Dibujo de repetición (Draw loop) y patrón de contador, los estudiantes escriben programas que mueven los sprites a través de la pantalla, así como también animan otras propiedades de los sprites.
Esta lección combina el dibujo con ciclos que los estudiantes conocieron anteriormente, de igual manera que el patrón de contador que aprendieron para crear programas con un movimiento determinado. Al aumentar o disminuir las propiedades de los sprites, como sprite.x, puedes escribir programas que muevan los sprites en los patrones esperados, en lugar de uno aleatorio que usamos en el pasado. Las animaciones que los estudiantes aprenden a crear en esta lección sientan las bases para todas las animaciones y juegos que harán en el resto del curso.

Secuencia para el aprendizaje

Conocimiento inicial (5 min)
Ampliación del conocimiento (40 min)
Transferencia del conocimiento (5 min)

Objetivos

Los estudiantes serán capaces de:

- Usar el patrón de contador para incrementar o disminuir las propiedades de los sprites.
- Identificar qué propiedades de los sprites deben cambiarse, y de qué manera, para lograr un movimiento específico.

Para los profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos
- El patrón de contador – Recurso
- Para los estudiantes
- Movimiento de sprites-Video

Estrategia de aprendizaje

Conocimiento inicial (5 min)

Revisando las propiedades de Sprites

Preguntar:

En una hoja de papel, enumera todas las propiedades de los sprites que puedas imaginar y el aspecto de un sprite que afecten.

Comente:

¿Qué tipo de animaciones podrías hacer al combinar las propiedades de los sprites con el patrón contrario? Considera agregar y restar propiedades, o incluso actualizar varias propiedades al mismo tiempo. Registra ideas mientras los estudiantes las comparten en el pizarrón.

El propósito de esta discusión es motivar a los estudiantes a pensar cómo podrían usar las diversas propiedades de sprites que han visto hasta ahora para hacer animaciones con un movimiento determinado. Si los estudiantes luchan para encontrar ideas, pueden limitar la pregunta a propiedades específicas. Por ejemplo:

- ¿Qué le pasaría a un sprite si aumentara constantemente su x propiedad?
- ¿Qué le pasaría a un sprite si aumentara constantemente su y propiedad?

¿Qué pasa con otras propiedades o combinando propiedades múltiples?

Ampliación del conocimiento (40 min)

Niveles: Sprites e imágenes

Transición:

Envía estudiantes a Code Studio - Movimiento Sprite.

Apoyo:

Los estudiantes deben progresar a través de los niveles sin detenerse en esta clase. Cuando hayan terminado los niveles de desarrollo de habilidades, se les dará la opción de qué proyecto desean extender.

Niveles de Code Studio:

- El patrón de contador
- Movimiento con el patrón de contador
- Depuración con vigilantes
- Animar a los Sprites
- Crea tu propia animación
- Desafíos
- Juego gratis

Transferencia del conocimiento (5 min)

Preguntar

Pida a los estudiantes que reflexionen sobre el desarrollo de las cinco prácticas del curso DISCOVERIES CS (Resolución de problemas, Persistencia, Creatividad, Colaboración, Comunicación). Elija una de las siguientes indicaciones según lo consideres apropiado.

- Elige una de las cinco prácticas en las que crees que demostraste crecimiento en esta Lección. Escribe algo que hiciste que ejemplifica esta práctica.
- Elige una práctica en la que pienses que puedes seguir creciendo. ¿Qué te gustaría mejorar?
- Elige una práctica que pensaste que era especialmente importante para la Lección que completamos hoy. ¿Qué lo hizo tan importante?

Lección 14: Booleanos desconectados

Lección sin conexión

Propósito

En esta lección, se les presenta a los estudiantes los valores booleanos y la lógica, así como las declaraciones condicionales. La clase comienza jugando un juego simple de levantarse (Stand Up), sentarse (Sit Down) en el que las afirmaciones booleanas (verdadero / falso) describen propiedades personales (color de pelo u ojos, tipo de ropa, edad, etc.). Esto hace que los estudiantes piensen cómo pueden enmarcar una propiedad con múltiples valores potenciales (como la edad) con una pregunta binaria.
A partir de ahí, los estudiantes reciben un grupo de objetos con propiedades físicas similares, pero variables. Con un compañero, agrupan esos objetos basándose en declaraciones booleanas cada vez más complejas, incluidos booleanos compuestos con AND (y) y OR (o).
Finalmente, revelamos que Condicionales es una herramienta para tomar decisiones o afectar el flujo de un programa que utiliza declaraciones booleanas como entrada.

Secuencia para el aprendizaje

Conocimiento inicial (10 min)
Ampliación del conocimiento (30 min)
Transferencia del conocimiento (5 min)

Objetivos

Los estudiantes serán capaces de:

- Organizar objetos basados en declaraciones booleanas simples y compuestas.
- Describir las propiedades de un objeto usando declaraciones booleanas.

Preparación

- Propiedades booleanas - Guía de actividades para cada estudiante.
- (Opcional) Reúne objetos con características similares pero variables para usar en lugar de la hoja de trabajo (los ladrillos LEGO funcionan bien, una bolsa de dulces mixta también puede ser divertida).

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los estudiantes:

- Propiedades booleanas - Guía de actividades.

Vocabulario

- **Booleano:** Un valor único de VERDADERO o FALSO.
- **Condicionales:** Declaraciones que solo se ejecutan bajo ciertas condiciones.
- **Expresión:** Cualquier unidad de código válida que se resuelve en un valor.

Estrategia de aprendizaje

Conocimiento inicial (10 min)

Párate, siéntate

Distribuir:

Entregue a cada alumno una tarjeta y pida que respondan las siguientes preguntas (puedes agregar algunas propias).

1. ¿Cuál es tu color de cabello?
2. ¿Usas lentes o lentes de contacto?
3. ¿Cuál es tu número favorito?
4. ¿Cuál es tu color favorito?
5. ¿En qué mes naciste?
6. ¿Tienes hermanos?
7. ¿Cuál es el último dígito de tu número de teléfono?
8. Escribe algo acerca de ti que las personas no saben y no pueden deducir con solo mirarte.

Luego recoja las cartas. Para jugar, sigue estos pasos:

Para cada tarjeta:

1. Selecciona una tarjeta
2. Di: Voy a leer la respuesta al # 7, pero si eres tú, no digas nada.
3. Lee la respuesta al # 7
4. Di: Ahora todos se ponen de pie y vamos a hacer algunas preguntas con respuestas booleanas para ayudar a determinar quién es esta persona. Voy a decir un montón de declaraciones. Si son ciertos, mantente en pie. Si son falsos, siéntate.
5. Traduce las respuestas del # 1 al # 7 en enunciados que pueden ser verdaderos o falsos, consulta a continuación, en voz alta. La persona que queda en pie es aquella que respondió verdadero a todas las afirmaciones y, por lo tanto, es responsable de la tarjeta.
6. Según cómo se preguntaron los números 3, 4, 5 y 7, es probable que más de una persona siga en pie. Si es el caso, vuelve a preguntarlas, esta vez restringiendo más la información, por ejemplo, “Mi color favorito es el violeta” para la pregunta 4.

Traducción de ejemplo

Juegue esto varias veces cambiando los enunciados “verdadero / falso” que usas. Sea creativo con el uso y recuerde a los estudiantes que la

Pregunta	Responder	Declaración verdadera / falsa
1) ¿Cuál es tu color de pelo?	marrón	"El color de mi cabello es marrón".
2) ¿Usa lentes o lentes de contacto?	sí	"Uso gafas o lentes de contacto".
3) ¿Cuál es tu número favorito?	12	"Mi número favorito es mayor que 10 y menor que 20."
4) ¿Cuál es tu color favorito?	púrpura	"Mi color favorito aparece en una puesta de sol".
5) ¿En qué mes naciste?	Mayo	"Nací en la primavera".
6) ¿Tienes hermanos?	sí	"Tengo hermanos".
7) ¿Cuál es el último dígito de su número de teléfono?	5	"El último dígito de mi número de teléfono es excelente".

OR (o) significa que una parte de la declaración es verdadera dará como resultado que toda la declaración sea verdadera.

Discute:
El juego Pararse/Sentarse con los estudiantes:

- ¿Qué tipo de cuestionamientos realizamos?
- ¿Alguna vez te confundiste sobre si debieses estar parado o sentado? ¿Por qué?
- En cualquier punto del juego, ¿en cuántos estados diferentes podrías estar?

Introduzca el vocabulario booleano como una descripción para los tipos de preguntas que estábamos haciendo. La característica que define a un booleano es que puede tener solo dos estados: en nuestro juego, esos estados eran Verdadero y Falso, o de pie y sentado.

Ampliación del conocimiento (30 min)

Hacer las preguntas correctas

Idea Genial

Preguntar:
Haga una lluvia de ideas sobre lugares donde hayan visto valores booleanos anteriormente, ya sea en la clase o en el mundo.
Discute:
Haga que los estudiantes compartan sus respuestas. Las posibles respuestas podrían incluir:

- Binario
- Diagramas de flujo
- Interruptores de luz (y otros dispositivos que pueden estar encendidos o apagados)

Ordenando con booleanos
Observaciones:
En el juego anterior, las preguntas booleanas que hice se basaron en sus propiedades. Sus propiedades no tienen que existir solo en dos estados (¿cuántos colores de cabello diferentes hay en el salón?), Pero las preguntas que hice tenían que dividirlas en dos estados (¿cuántas personas en el salón tienen el pelo rojo?). Vamos a hacer una clasificación similar usando las propiedades de varias imágenes.

Los estudiantes a menudo luchan con la idea de hacer *mayor que* o *igual a* sersiempre FALSO - tienden a pensar en esos como declaraciones de verdad enlugar de preguntas de relación.
En esta Lección usamos el lenguaje *algo* es igual a *otra cosa* para reflejar el código que verán más adelante (porejemplo something == something_else), pero para los estudiantes que estan luchando por ver esto como unapregunta en lugar de una afirmación, puede alentarlos a que reformulen la afirmación moviendo el 'es' al principio, entonces:
algo es igual a otra cosa
se convierte
¿es algo igual a otra cosa?

Al combinar booleanos con " O", estamos utilizando lo que se llama *lógica o* - lo que significa que estamospreguntando si alguno (o ambos) o los booleanos son VERDADEROS. A menudo, los estudiantes piensan en O como *exclusivo o* - lo que significa que estamos preguntando si solo uno (pero no el otro) es VERDADERO.
Asegúrese de hacer preguntas o preguntarles a los estudiantes donde losdos booleanos son VERDADEROS para asegurarse de que pueda obtener estaidea equivocada temprano.

Grupo:
Organiza a los estudiantes en parejas.
Configuración:
Asigna a cada miembro del par como verdadero o falso.
Distribuir:
Distribuye las imágenes recortadas de la hoja de trabajo o proporciona a los estudiantes algunos objetos para ordenar.
Observaciones:
Voy a leer un montón de enunciados binarios en forma de "la figura equivale a un cuadrada o tiene más de 4 lados", y ustedes van a clasificar sus objetos para organizarlos en pilas VERDADERAS y FALSAS.
Si los estudiantes no están de acuerdo con respecto a qué pila debe colocar un objeto, primero deben analizar cuál es la propiedad, cuáles son los dos resultados de la pregunta binaria y luego, si aún no pueden ponerse de acuerdo, deben llevarla a la clase para votar.
Si está utilizando los recortes proporcionados, puede comenzar con las siguientes preguntas:

- Lados es igual a 3
- El relleno es igual a negro
- Las esquinas son menos de 1
- El ancho es igual a la altura
- El relleno es igual a gris Y lados son más que 4
- Lados es mayor que 4 y menos de 7
- Lados es mayor que 4 o menor que 7
- Lados es más que o igual a 5

Condicionales
Meta:

Después de acostumbrarnos a ordenar objetos en VERDADERO y FALSO, debemos presentarles a los estudiantes el concepto de que los Booleanos también se pueden usar para controlar el flujo de un programa.
Observaciones:
Una condicional nos permite tomar una decisión basada en el resultado de una pregunta (o condición) booleana. De hecho, estábamos usando implícitamente condicionales en la Lección de Pararse/Sentarse porque había una acción relacionada con cada resultado potencial del booleano. Podríamos haber reformulado las instrucciones como si la declaración es verdadera: permanece de pie de lo contrario; siéntate.
Preguntar:
Selecciona un objeto de tu montón y sostenlo.
Observaciones:
Voy a hacerles una pregunta booleana sobre su objeto y darles una acción relacionada con el resultado del booleano. Averigüe cuál debería ser su respuesta para su forma y haga la respuesta correcta.
Preguntar:
Haga a los estudiantes algunas preguntas booleanas sobre ese único objeto y deles algo que hacer si esa pregunta es verdadera. Por ejemplo:

- Si los lados son iguales a 4, haz un baile.
- Si el patrón es igual a rayado, siéntate.
- Si el ancho es igual a la altura, salta sobre un pie.

Transferencia del conocimiento (5 min)

Condición explícita

Objetivo:
Los booleanos y condicionales son en realidad algo que utilizamos en nuestra vida cotidiana, simplemente no solemos ser explícitos al respecto.
Modelo:
Como una forma de practicar el pensamiento de forma explícita sobre condicionales, considera despedir a tus estudiantes utilizando compuestos booleanos y condicionales. Por ejemplo.

- Si te sientas en la mesa cuatro y tu cabello es marrón, puedes irte.
- Si tu primer nombre comienza con A o B, puedes irte.
- Si tus zapatos son negros, puedes irte.

Sugerencias para evaluación

Se sugiere el siguiente indicador para evaluar formativamente los aprendizajes:

- Organizan objetos en base a criterios simples
- Describen el proceso de trabajo para la organización de los elementos con los que trabajan.

Lección 15: Booleanos y condicionales

Lección en línea Ver en Code Studio

Propósito

Los estudiantes comienzan usando booleanos para comparar el valor actual de una propiedad de sprite con un valor objetivo, usando esa comparación para determinar cuándo un sprite ha alcanzado un punto en la pantalla, crecido a un tamaño determinado o alcanzado un valor usando el patrón contrario. Después de usar booleanos directamente para investigar los valores o las propiedades de sprite, los estudiantes agregan sentencias condicionales "if" (si) para escribir código que respondan a esas comparaciones booleanas.
Esta lección sigue de cerca el modelo booleano que los estudiantes experimentaron por primera vez en la lección Booleanos desconectada. Como antes, comenzamos con el uso de booleanos directamente antes de usar booleanos para desencadenar sentencias if. En la siguiente lección, presentaremos algunos bloques de producción booleana, como keyDown(), que se pueden usar en lugar de comparaciones booleanas simples para escribir programas que respondan a la entrada del usuario.

Secuencia para el aprendizaje

Conocimiento inicial (5 min)
Ampliación del conocimiento (40 min)
Transferencia del conocimiento (5 min)

Objetivos

Los estudiantes serán capaces de:

- Predecir el resultado de declaraciones booleanas simples.
- Usar condicionales para reaccionar a los cambios en las variables y las propiedades de los sprites.

Para los profesores:

- Operadores booleanos y de comparación - Recurso
- CSD Unidad 3 - Animaciones y Juegos Interactivos
- Sentencias si - Recurso

Para los estudiantes:

- Expresiones booleanas – Video
- Sentencias condicionales – Video

Vocabulario

- **Expresión Booleana:** En la programación, una expresión que se evalúa como Verdadera o Falsa.
- **If-Statement:** La estructura de programación común que implementa "declaraciones condicionales".

Código

- If statement
- Equality operator
- Inequality operator
- Greater than operator
- Greater than or equal operator
- Less than operator
- Less than or equal operator

Estrategia de aprendizaje

Conocimiento inicial (5 min)

Contestando preguntas Booleanas

Meta:
Al final del juego, con las preguntas booleanas de la lección anterior, los estudiantes comenzaron a agregar condiciones a sus preguntas booleanas. Significa que, si la respuesta a la pregunta es verdadera, debería suceder algo. Antes de programar con condicionales, queremos asegurarnos de que los estudiantes tengan un conocimiento sólido de lo que realmente son los booleanos.
Rápido:

- ¿Cuántos números diferentes hay en el mundo?
- ¿Cuántas palabras diferentes o combinación de letras y otros personajes hay?
- ¿Cuántos valores booleanos diferentes hay?

Discutir:
Los estudiantes deben darse cuenta de que las dos primeras preguntas (números y cadenas) son esencialmente infinitas, pero que los booleanos están limitados a dos estados.
Observaciones:
A medida que comience a programar hoy, estará usando booleanos para hacer programas que cambien su comportamiento dependiendo de la respuesta a esas preguntas booleanas.

Ampliación del conocimiento (40 min)

Booleanos

Transición:
Envía a los estudiantes a Code Studio y realice los Niveles de Code Studio - Condicionales.

- Booleanos y operadores de comparación
- Comparación Booleana
- Declaraciones “si”
- Condicionales Básicos
- Juego gratis

Transferencia del conocimiento (5 min)

Agregar condicionales

Reflexión:
Piense en todos los programas que ha escrito hasta ahora;
¿Cómo podrías usar condicionales para mejorar uno de tus programas de actividades pasadas? ¿Qué condición verificaría y cómo respondería a ella?

Sugerencias de evaluación

Se sugiere el siguiente indicador para evaluar formativamente los aprendizajes:

- Representan con ejemplos simples la lógica booleana
- Combinan diversos criterios de búsqueda utilizando los operadores booleanos
- Incorporan preguntas condicionales en la construcción de sus soluciones

Aunque aparentemente es simple, entender como una declaración booleana evaluará puede ser difícil dado que los diferentes lenguajes de programación tienen opiniones diferentes sobre “veracidad” y “falsedad”. De hecho, JavaScript (el lenguaje utilizado en este curso) tiene dos operadores diferentes para probar la igualdad booleana == y ===.
El operador doble igual (==) es bastante generoso para determinar la veracidad, por ejemplo, cada uno de los siguientes se considera true (verdadero) en JavaScript cuando se usa el == operador, pero se usaría el === operador: para false
1==”1” verdadero
1===”1” falso

1 == verdadero;
“1” == verdadero;
5 == “5”;
null == undefined; “ “ == falso;

Usamos el == operador en este curso porque es más indulgente, pero es importante tener en cuenta que a veces puede informar la verdad cuando en realidad no tenía la intención de hacerlo (en cuyo caso podría querer usar el === operador más estricto)

Lección 16: Condicionales y entrada del usuario

Lección en línea Ver en Code Studio

Propósito

Después de la introducción a las declaraciones booleanas y si en la Lección anterior, se introduce a los estudiantes en un nuevo bloque llamado keyDown() que devuelve un booleano y se puede usar en sentencias de condicionales para mover sprites alrededor de la pantalla. Al final de esta lección, los estudiantes tendrán programas escritos que tomarán la entrada de teclado del usuario para controlar los sprites en la pantalla.
Una forma común de usar condicionales es verificar los diferentes tipos de entrada del usuario, especialmente las pulsaciones de teclas. Tener una forma para que un usuario interactúe con un programa, lo hace más interesante y dinámico. Sin interacción del usuario, es muy difícil crear un juego. Por lo tanto, la introducción de condicionales y aportes de los usuarios para la toma de decisiones es el primer gran paso hacia la creación de juegos.

Secuencia para el aprendizaje

Conocimiento inicial (5 min)
Ampliación del conocimiento (40 min)
Transferencia del conocimiento (5 min)

Objetivos

Los estudiantes serán capaces de:

- Usar condicionales para reaccionar a la entrada del teclado
- Mover los sprites en respuesta a la entrada del teclado.

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos

Código

- keyDown(code)

Estrategia de aprendizaje

Conocimiento inicial (5 min)

Tomando entrada

Debate:

Hasta ahora, todos los programas que ha escrito se ejecutan sin intervención del usuario. ¿De qué manera la adición de la interacción del usuario puede hacer que sus programas sean más útiles, efectivos o entretenidos? ¿Cómo podría un usuario proporcionar información sobre su programa?

El objetivo aquí no es entrar en las especificaciones técnicas de cómo los programas pueden recibir información (los estudiantes entenderán eso en la parte en línea de la Lección), sino más bien hacer que los estudiantes piensen en cómo permitir que los usuarios que ingresen puedan cambiar los programas que ellos 'han hecho'. Anime a los estudiantes a pensar en las diversas entradas y salidas informáticas que existen ¿Qué entradas serían más útiles para los tipos de programas que han estado haciendo?

Ampliación del conocimiento (40 min)

Entrada de teclado

Transición:

Enviar los estudiantes a Code Studio, contenido de Lección

- Entrada de teclado
- Edición de imágenes
- Desafío: Agregar más entradas

Transferencia del conocimiento (5 min)

Considerando las condiciones

Indicación:

Para que los estudiantes continúen pensando en cómo se pueden usar los condicionales en la programación, pida que ideen escenarios en juegos o programas que usan regularmente que pueden ser activados por los condicionales.

Discute:

Haga que los estudiantes compartan las respuestas. Las respuestas de los estudiantes pueden incluir:

- Si mi nombre de usuario y contraseña son correctos, conéctate a Facebook
- Si Pacman ha reunido todas las bolas, comienza el siguiente nivel
- Si mi teclado o mouse no se movió en 10 minutos, encienda el protector de pantalla.

Sugerencias de evaluación

Se sugiere el siguiente indicador para evaluar formativamente los aprendizajes:

- Usan sentencias condicionales en un proyecto de programación,
- Permitir que los usuarios que interactúen con el proyecto puedan describir la lógica programada.

Lección17: Otras formas de entrada

Lección en línea Ver en Code Studio

Propósito

En esta lección, los estudiantes continúan explorando maneras de usar declaraciones condicionales para tomar la opinión del usuario. Además del comando keyDown() simple aprendido ayer, los estudiantes aprenderán sobre varios otros comandos de entrada de teclado, así como formas de tomar la entrada del mouse.
Los estudiantes han aprendido cómo tomar decisiones simples con condicionales. A veces, sin embargo, queremos tomar una decisión en función de si la condición sobre la que preguntamos originalmente era falsa o si queremos tomar una decisión basada en que múltiples condiciones sean ciertas. Ahí es, donde aparecen enunciados y condicionales más complejos. Estas declaraciones son una segunda declaración adjunta a una instrucción if (si). Otras declaraciones se ejecutan cuando la sentencia if (si) a la que está asociada es falsa. Puedes pensarlo como “si algo es verdadero haz cosa 1 sino (else) haz cosa 2”.
Este concepto se presenta junto con varios comandos nuevos de ingreso de teclas y mouse, lo que permite a los estudiantes crear gradualmente programas que ingresan de diferentes maneras.

Secuencia para el aprendizaje

Conocimiento inicial (5 min)
Ampliación del conocimiento (40 min)
Transferencia del conocimiento (5 min)

Objetivos

Los estudiantes serán capaces de:

- Usar una instrucción else (sino) como el caso de respaldo a una sentencia if (si).
- Diferenciar entre las condiciones que son verdaderas una vez por interacción y aquellas que permanecen verdaderas a lo largo de la duración de una interacción.

Para los profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos
- Sentencias si-si no – Recurso

Para los estudiantes:

- Sentencias `si/si no` - Vídeo

Vocabulario

- **Condicionales:** declaraciones que solo se ejecutan bajo ciertas condiciones.

Código

- keyWentDown(code)
- keyWentUp(code)
- mouseDidMove()
- mouseDown(button)
- mouseWentDown(button)
- MouseWentUp(button)
- sprite.visible
- If/else statement

Estrategia de aprendizaje

Conocimiento inicial (5 min)

Verificar la comprensión

Observaciones

Hoy retomaremos donde lo dejamos en la lección anterior, usando condicionales para escribir programas que respondan a los comentarios de los usuarios. Actualicemos lo que aprendimos.

Rápido:

- ¿Qué es un booleano? (respuesta; un valor verdadero / falso)
- ¿Cuál es la relación entre un booleano y un condicional? (resp; un condicional hace una pregunta booleana y ejecuta el código si la respuesta es verdadera)
- ¿Cuáles son algunos ejemplos de operadores de comparación que dan como resultado un booleano? (resp.,>, <, ==)
- ¿Cuál es la diferencia entre = y ==? (resp., =se usa para asignar un valor, == se usa para verificar si dos valores son iguales).

Ampliación del conocimiento (40 min)

Sí / más y más entradas

Video:

Mire el video de condicionales (conditional) juntos como una clase. Será una revisión de las declaraciones if (si) e introducirá también algunos conceptos nuevos.

Transición:

Enviar estudiantes a Code Studio Contenido de la Lección

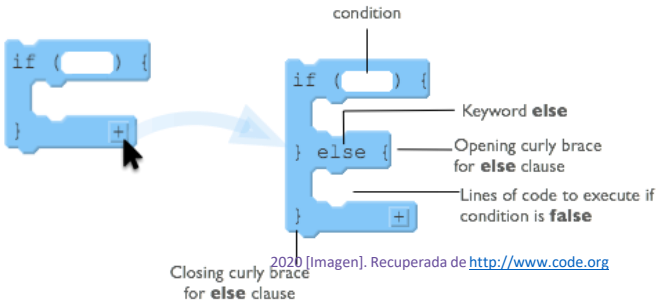
- Sigue al mouse
- Video: Condicionales
- Declaraciones de If-Else (Si-Sino)
- Entrada con If-Else (Si-Sino)
- Desafío: verificar múltiples condiciones

If-Else (Si-Sino) Declaraciones

Cómo funcionan las declaraciones If-Else (Si-Sino)

Con una instrucción if-else, usted está dando un comando cualquiera-or (o): en el cual se ejecutarán las líneas de código dentro de if o las líneas dentro de else. Esas son las opciones.

En el video (**recuerde activar los subtítulos**) se muestra cómo agregar una cláusula else a una instrucción if: pulsa el pequeño símbolo + en la cola de la declaración if.

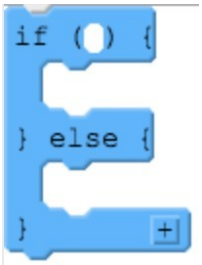


2020 [Imagen]. Recuperada de http://www.code.org

Dentro de las llaves de la cláusula else, coloque las líneas de código que deseas ejecutar si la condición booleana de la declaración if es falsa.

Algunas notas importantes sobre la cláusula else:

- El else debe venir inmediatamente después de la llave de cierre de una declaración if.
- El otro else también tiene su propio conjunto de llaves de apertura y cierre para encapsular líneas de código.
- ¿Qué pasa si mi condición no es verdadera?



A veces queremos decirle a nuestro programa qué hacer si se trata de una condición true, pero también qué hacer si es así false. Al presionar el botón más en la parte inferior de su bloque condicional le dará otra sección llamada else. Esta sección else es una alternativa: se llamará siempre que exista la condición if anterior que sea falsa.

Clics del mouse

Las pulsaciones de teclas son geniales, pero a veces desea que los usuarios interactúen a través de los clics del mouse. Hay un nuevo bloque llamado mouseDown() que, similar a keyDown(), comprueba si se presionan los botones izquierdo o derecho del mouse. Si está usando una computadora con un mouse o un panel táctil que solo tiene un botón, querrá usar siempre mouseDown("left").

mouseDidMove (el mouse se movió)

También podemos usar expresiones booleanas para verificar si el mouse se movió o no. El bloque mouseDidMove devolverá false si el mouse está quieto, pero true si el mouse se ha movido.

Transferencia del conocimiento (5 min)

Compartir

el último programa de esta Lección consiste en pedir a los estudiantes que amplíen un programa a usos condicionales para la participación del usuario de maneras nuevas e interesantes. Haga que sus estudiantes compartan sus proyectos con la clase, lo que les pedirá que expliquen cómo usaron los condicionales en sus programas.

Sugerencias de evaluación

Se sugieren los siguientes indicadores para evaluar formativamente los aprendizajes:

- Crean variables con nombres claros que expliquen tipos de datos
- Incorporan condicionales y explican la funcionalidad de forma predictiva
- Prueban y perfeccionan sistemáticamente los programas utilizando una serie de casos de prueba

Lección 18: Proyecto – Tarjeta interactiva

Lección en línea Ver en Code Studio

Propósito

En este proyecto, los estudiantes planifican y desarrollan una tarjeta de felicitación interactiva usando todas las técnicas de programación que han aprendido hasta este momento.
Esta evaluación es un buen lugar para que los estudiantes junten todas las piezas que aprendieron (dibujo, variables, sprites, imágenes, condicionales, entrada del usuario) en un solo lugar.
Los estudiantes todavía deberían estar trabajando con un código que sea fácil de leer y que no sea difícil de entender para ellos.
Darles a los estudiantes la oportunidad de ser realmente creativos después de aprender todos estos nuevos conceptos los ayudará a involucrarse más en el contenido.

Secuencia para el aprendizaje

Conocimiento inicial (10 min)
Ampliación del conocimiento (2 días)
Transferencia del conocimiento (10 min)
Evaluación

Objetivos

Los estudiantes serán capaces de:

- Usar condicionales para reaccionar a la entrada del teclado o cambios en las variables / propiedades.
- Usar Comandos de secuencia para dibujar en el orden correcto.
- Aplicar un patrón de iterador a las variables o propiedades en un ciclo.

Preparación

- Tarjeta Interactiva - Guía de actividades

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos
- Código Extra en los Niveles de Desafío – Recurso

Para los estudiantes:

- Prácticas de las ciencias de la computación - Reflexión
- Tarjeta interactiva – Guía de Proyecto
- Tarjeta interactiva – Revisión por pares
- Tarjeta interactiva – Rúbrica
- Tarjetas interactivas - Lista de Comprobación del Estudiante

Estrategia de aprendizaje

Conocimiento inicial (10 min)

Meta: Los estudiantes ven un ejemplo de un proyecto final y discuten los diferentes elementos que se utilizaron para hacerlo.

Demo Ejemplo de proyecto (Nivel 2)

Pantalla:
En el proyector, ejecute el ejemplo para los estudiantes. Dado que esto está en el nivel 2 del progreso, si es más fácil para los estudiantes hacerlo en sus propias computadoras, también podría hacer eso.

Niveles de Code Studio

Ejemplo de tarjeta interactiva CSD U3

Proyecto de ejemplo

Ejecute el programa varias veces y responda las siguientes preguntas:

1. ¿Qué elementos parecen usar comandos de dibujo?
2. ¿Qué elementos parecen ser Sprites?
3. Para cada Sprite, ¿qué propiedades se están actualizando?
4. ¿Dónde ves condicionales que se utilizan?
5. ¿Hay elementos que no entiendes?

Discute:
Haga que los estudiantes compartan sus observaciones y análisis del ejemplar.
Alienta a la clase a considerar que existen múltiples enfoques para programar cualquier cosa, pero que puede haber pistas sobre cómo se creó algo. En particular, cuando comparten sus ideas, pida que especifiquen lo siguiente:

- Pistas que sugieren que se utilizó un Sprite
- Se usaron pistas que sugieren un condicional
- Se usaron pistas que sugieren un patrón de iterador

Pantalla:
Muestra a los estudiantes la tarjeta interactiva - rúbrica. Revisa los diferentes componentes de la rúbrica con ellos para asegurarse de que entienden los componentes del proyecto.

Ampliación del conocimiento (2 días)

Meta: Los estudiantes deben planear lo que quieren crear antes de dirigirse a la computadora para que, una vez que lleguen a ella, simplemente estén ejecutando el plan.

Desconectado: planificación interactiva de tarjetas

Distribuir:
Entregue la Tarjeta Interactiva - Guía de actividades para los estudiantes. Ésta es la herramienta que los estudiantes usarán para determinar sus proyectos antes de ingresar a las computadoras. Deles tiempo para que hagan una lluvia de ideas sobre el tipo de tarjeta que quieren crear y quién será el destinatario.

Pasos

1. La primera capa de la tarjeta interactiva es un fondo dibujado con sólo los comandos en el cajón de Dibujo. El anverso de la Guía de actividades proporciona una cuadrícula para que los estudiantes expongan sus fondos, una tabla de referencia de comandos de dibujo y un área para que los estudiantes tomen notas y escriban pseudocódigos.
2. Los siguientes estudiantes piensan en los Sprites que necesitarán, completando una tabla con la etiqueta, las imágenes y las propiedades de cada Sprite.
3. Finalmente, los estudiantes consideran los condicionales que necesitarán para hacer su tarjeta interactiva.

Niveles: implementación de la tarjeta interactiva (nivel 3 - 7)

Transición:
Una vez que los estudiantes hayan completado su hoja de planificación, es hora de dirigirse a las lecciones de Code studio. La secuencia de nivel corto les pide a los estudiantes completar cada elemento de su proyecto.

Revisión por pares

Distribuir:
Entregue a cada alumno una copia de la Tarjeta interactiva - Revisión por pares.
Los estudiantes deben pasar 15 minutos revisando la tarjeta del otro alumno y completando la guía de revisión por pares.

Iterar - Actualizar código

Recorra la sala:
Los estudiantes deben completar la guía de la revisión por pares y decidir cómo responder a los comentarios que se les dieron. Luego deberían usar ese comentario para mejorar sus cartas.

Reflexionar

Usando la Tarjeta interactiva - Rúbrica, los estudiantes deben evaluar su propio proyecto antes de enviarlo.
Envíe a los estudiantes a Code Studio para completar su reflexión sobre sus actitudes hacia la informática. Aunque sus respuestas son anónimas, los datos agregados estarán disponibles una vez que al menos cinco estudiantes hayan completado la encuesta.

Transferencia del conocimiento (10 min)

Compartir tarjetas

Meta: Los estudiantes comparten sus creaciones con la clase.
Compartir:
Encuentre una forma para que los estudiantes compartan sus tarjetas entre ellos y con el destinatario deseado. Es probable que sea útil utilizar el enlace compartido para el proyecto, de modo que los estudiantes puedan compartir el proyecto con otros estudiantes.

Evaluación

Se proporciona una rúbrica para evaluar los proyectos de los estudiantes en los recursos.

Unidad 1 - ¿Qué es el pensamiento computacional y la programación?

Contenido 3 - Creación de videojuegos

Resumen

7 actividades:

- Lección 19: Velocidad
- Lección 20: Detección de colisión
- Lección 21: Movimiento complejo de Sprite
- Lección 22: Colisiones
- Lección 23: Funciones
- Lección 24: El proceso de diseño del juego
- Lección 25: Uso del proceso de diseño del juego

Objetivos

- OA 1. Aplicar conceptos de Ciencias de la Computación –abstracción, organización lógica de datos, análisis de soluciones alternativas y generalización– al crear el código de una solución computacional.
- OA g. Elaborar representaciones, tanto en forma manual como digital, y justificar cómo una misma información puede ser utilizada según el tipo de representación.
- OA a. Construir y evaluar estrategias de manera colaborativa al resolver problemas no rutinarios.
- OA d. Argumentar, utilizando lenguaje simbólico y diferentes representaciones para justificar la veracidad o falsedad de una conjetura, y evaluar el alcance y los límites de los argumentos utilizados.

Referencias

- Code Studio - Code.org
- Cuantrix - Fundación Televisa
- CSTA – Computer Science Teachers Association

Lección 19: Velocidad

Lección en línea Ver en Code Studio

Propósito

Después de una breve revisión de cómo utilizaron el patrón de contador para mover los sprites en las lecciones anteriores, los estudiantes reciben las propiedades que establecen la velocidad y rotación. A medida que utilizan estas nuevas propiedades de diferentes maneras, desarrollan las habilidades necesarias para crear un juego básico de desplazamiento lateral.
En esta Lección, se les enseña a usar los bloques de velocidad para simplificar el código para mover un sprite a través de la pantalla. Esto marca un cambio en cómo se introducen los nuevos bloques. Mientras que los bloques anteriores se presentaron como habilitantes de comportamientos completamente nuevos, ahora se presentan como un código simplificador que los estudiantes podrían haber escrito con los bloques disponibles anteriormente. En las siguientes lecciones, los estudiantes verán cómo este método les permite producir comportamientos de sprites más interesantes.

Secuencia para el aprendizaje

Conocimiento inicial (15 min)
Ampliación del conocimiento (75 min)
Transferencia del conocimiento (5 min)

Objetivos

Los estudiantes serán capaces de:

- Usar los bloques velocidad y rotación para crear y cambiar los movimientos de los sprites.
- Describir las ventajas de simplificar el código mediante el uso de bloques de mayor nivel.

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos
- Velocidad – Recurso
- Para los estudiantes:
- Velocidad – Video

Código

- sprite.rotationSpeed
- sprite.velocityX
- sprite.velocityY

Estrategia de aprendizaje

Conocimiento inicial (15 min)

Demuestre:
Pida que un voluntario pase al frente de la clase y actúe como su sprite. Digamos que le dará instrucciones al sprite como si fuera un programa de Game Lab.
Cuando su estudiante esté listo, organice el lugar para que tengan algo de espacio frente a ellos y pida que "avancen por 1". Deben dar un paso adelante. Luego repita el comando varias veces, cada vez que esperas que el estudiante avance un paso. Debería apuntar a que la repetitividad de estas instrucciones sea clara. Después de que tu estudiante haya completado esta Lección, pide que vuelvan a donde comenzaron. Esta vez repite la demostración, pero solicita al alumno que "Avance por 2" y haga que el alumno tome 2 pasos cada vez. Una vez que el alumno haya hecho esto varias veces, pida a la clase que dé una ronda de aplausos e invita a los estudiantes volver a su asiento.
Preguntar:
Sólo estaba dando instrucciones a mi Sprite, pero parecían ser bastante repetitivas. ¿Cómo podría haber simplificado o simplificado mis instrucciones?
Discute:
Dé a los estudiantes un minuto para escribir sus pensamientos antes de invitarlos a compartir con un compañero. Luego haga que la clase comparta sus pensamientos. Puede escribir sus ideas en el pizarrón.
Observaciones
Una forma de simplificar estas instrucciones es decirle a nuestro sprite que siga moviéndose por 1 o 2, o por todos los pasos que queramos. Esto haría las instrucciones más fáciles de entender para los humanos, y como estamos a punto de ver, también hay una manera similar de simplificar nuestro código.
Muestra:
Video de Sprite Velocity
Pantalla:
En el pizarrón, escriba las siguientes piezas de código.

- sprite.velocityX = 4;
- sprite.velocityY = -1;
- sprite.rotationSpeed = 2;

Indicación:
Acabamos de ver cómo estos nuevos bloques nos ayudan a simplificar el patrón de contador que usamos para mover los sprites. En una hoja de papel, anote el patrón de contador que está reemplazando cada una de estas líneas de código.
Discute:
Haga que los estudiantes escriban sus ideas en una hoja de papel antes de hablar sobre sus respuestas como clase. Finalmente, escriba las respuestas correctas en el tablero al lado de cada una, como se muestra a continuación.

- sprite.x = sprite.x + 4;
- sprite.y = sprite.y - 1;
- sprite.rotation = sprite.rotation + 2;

Observaciones
Estos nuevos bloques de "nivel superior" nos ayudan a escribir código que ya sabíamos cómo escribir. Están simplificando nuestro código para nosotros al ocultar algunos de los detalles innecesarios. Sin embargo, como estamos a punto de ver, estos nuevos bloques cambiarán el tipo de programas y juegos que podemos escribir.

Meta: la demostración anterior debería haber reforzado el hecho de que repetirlas mismas instrucciones es algo que nunca harías en la vida real. En su lugar, se le ocurriría una forma de capturar que las instrucciones deberían repetirse, como "seguir avanzando por 1"

Comprobar la comprensión: esto es solo una comprobación rápida para entender después del video. Se les pide rápidamente a los estudiantes que vean si entendieron el punto principal del video y, si no, tienen la oportunidad de reforzarlo antes de pasar a los niveles de Code Studio. Deje la traducción delos bloques de velocidad a si patrón decontador asociado en el tablero para hacer referencia a lo largo de la Lección

Ampliación del conocimiento (75 min)

Aprendiendo a usar los bloques de velocidad
Transición:
Los estudiantes se dirigen a Code Studio, Contenido de la Lección -Code.org.

Recorra la sala:

¿Dentro o fuera del lazo de dibujo?: Para los primeros rompecabezas, asegúrese de que los estudiantesestén configurando las velocidadesy velocidades de rotación *fuera* del círculo de dibujo, inmediatamente después de que creen sus sprites. El código funcionará para los primeros rompecabezas, incluso si establecenlas velocidades dentro del bucle de extracción, pero causará problemas más adelante.

Estos niveles introducen las propiedades de velocidad X, velocidad Y velocidad de rotación que acaba de analizar con los estudiantes. Verifique con los estudiantes para ver cómo están y mantenga un registro de cuándo todos han llegado al final del nivel 10.

VelocidadX

Una forma de mover los sprites en Game Lab es con el patrón de contador. Por ejemplo, sprite1.x = sprite1.x + 1 mueve un sprite por 1 píxel en cada fotograma del ciclo de dibujo. Este patrón es tan común que los sprites tienen una velocityX, propiedad que hace esto por ti.

Velocidad de rotación

Ya ha aprendido cómo hacer que su sprite gire usando el rotation bloque. Por ejemplo, cuando quería que su sprite girara dos grados cada vez que se dibujaba, colocabas sprite.rotation = sprite.rotation + 2 dentro del círculo de extracción. Ahora puede usar los rotationSpeed para hacer que sus sprites giren una cierta cantidad cada vez que se dibujan. Si desea que su sol gire dos grados cada vez que se dibuja, puede usar sun.rotationSpeed = 2 antes del ciclo de extracción, después de crear su sprite.

Control de velocidad

Usó rotatationSpeed fuera del lazo de extracción para hacer que tu sprite girara cuando tu programa comienza. También puedes usar rotationSpeed dentro del ciclo de dibujo para cambiar la velocidad del sprite durante el juego. Por ejemplo, un Sprite puede comenzar a girar cuando el usuario presiona la barra espaciadora, y seguirá girando hasta que se le indique que se detenga.

Cambiando la velocidad con la posición

Una ventaja del uso de bloques de velocidad dentro de condicionales (if bloques) es que tu sprite se mantiene en movimiento, incluso después de que la condición deje de ser cierta.

Si los estudiantes terminan temprano, anímelos a experimentar con diferentes aspectos del juego de desplazamiento que crearon. Pueden cambiar las animaciones, crear un fondo, etc.

Transferencia del conocimiento (5 min)

Indicación:

Hoy aprendieron algunos bloques nuevos. A primera vista, estos bloques hicieron el mismo tipo de cosas que ya habíamos hecho con el patrón de contador, pero nos simplificaron hacerlas. Sin embargo, al pasar por los rompecabezas, comenzaron a hacer algunos movimientos interesantes que no habíamos podido hacer antes.

- Describe uno de esos movimientos y cómo lo hiciste.
- Describe otro movimiento que te gustaría hacer, pero aún no sabes cómo.
- Describe otro bloque que te gustaría tener.
- ¿Tomaría algún argumento?
- ¿Qué haría?
- ¿Qué código escondería dentro?

Observaciones

Todos los movimientos que hicimos hoy son posibles sin los nuevos bloques, pero sería muy complicado codificarlos. Uno de los beneficios de bloques como la velocidad es que cuando no tenemos que preocuparnos por los detalles de los movimientos y acciones simples, podemos usar esa potencia cerebral adicional para resolver problemas más complicados. A medida que desarrolle su juego de desplazamiento lateral, seguiremos buscando nuevos bloques que simplifiquen las cosas, para que podamos construir juegos cada vez más complicados.

Sugerencias para evaluación

Se sugieren los siguientes indicadores para evaluar formativamente los aprendizajes:

- Utilizan bloques de velocidad y rotación, cambiando el movimiento de los sprites.
- Incorporan el código, los medios y las bibliotecas existentes en los programas originales, y le dan atribución.
- Prueban y perfeccionan sistemáticamente los programas.
- Documentan aquellos programas para que sean más fáciles de seguir, probar y depurar.

Lección 20: Detección de colisión

Lección en línea Ver en Code Studio

Propósito

Los estudiantes aprenden sobre la detección de colisión en la computadora. Trabajando en parejas, exploran cómo una computadora puede usar la ubicación de sprites y propiedades de tamaño y matemáticas para detectar si dos sprites se están tocando. Luego usan el bloque isTouching() para crear diferentes efectos cuando los sprites colisionan, incluidos los sonidos de reproducción. Por último, usan sus nuevas habilidades para mejorar el juego de desplazamiento lateral que comenzaron en la última Lección.
Esta Lección introduce formalmente el uso de abstracciones, formas simples de representar la complejidad subyacente.
En la última Lección, los estudiantes fueron expuestos a la idea de usar un bloque para representar código complejo. Los estudiantes exploran más a fondo esta idea en el contexto del desafío matemático complejo e intencional de determinar si dos sprites se están tocando. Al usar un solo bloque para representar esta complejidad, en este caso, el bloque isTouching, se vuelve mucho más fácil escribir y razonar sobre el código, y los estudiantes pueden apreciar el valor de usar abstracciones. En lecciones posteriores, los estudiantes continuarán construyendo sobre la abstracción isTouching() para crear interacciones de sprites más complejas.

Secuencia para el aprendizaje

Conocimiento inicial (10 min)
Ampliación del conocimiento
Transferencia del conocimiento (5 min)

Objetivos

Los estudiantes serán capaces de:

- Utilizar el bloque isTouching para determinar cuándo se tocan dos sprites.
- Describir cómo las abstracciones ayudan a manejar la complejidad del código.

Preparación

- Imprima copias de la guía de actividades. Considere que cada pareja de estudiantes debe tener la parte A y B de la guía

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los Profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos
- Detección de colisiones - Recurso

Para los estudiantes:

- Detección de Colisión (Versión A) - Guía de Actividades
- Detección de Colisión (Versión B) - Guía de Actividades

Vocabulario

- **Abstracción:** una representación simplificada de algo más complejo. Las abstracciones le permiten ocultar detalles para ayudarlo a manejar la complejidad, enfocarse en conceptos relevantes y razonar sobre problemas en un nivel superior.
- **Depuración:** encontrar y solucionar problemas en un algoritmo o programa.
- **If-Statement**: la estructura de programación común que implementa “declaraciones condicionales”.

Código

- sprite.isTouching(target)
- sprite.debug

Estrategia de aprendizaje

Conocimiento inicial (10 min)

Revisión:
Recuerde brevemente a los estudiantes el juego de desplazamiento lateral que hicieron en la última Lección y pida que compartan cualquier idea que tengan para mejorar su juego en el futuro.

Pantalla:
En un proyector o simplemente en una pantalla de computadora, muestre el juego que aparece en el primer nivel en Code Studio para esta lección. El maestro o un alumno puede controlar a la rana para saltar sobre un hongo. Asegúrese de que al menos algunos saltos sean exitosos y que algunos no vean que la detección de colisiones está funcionando.

Mostrar el juego: en este punto,sólo muestre el juego. Tienen una actvidad desconectada bastante importante inmediatamente después y importante superarla antes de interactuar con el juego.

Inducción:
Una mejora interesante en este juego es que el hongo se mueve cuando la rana lo toca. ¿Puedes pensar en alguna forma en que la computadora pueda usar las propiedades de los sprites para averiguar si se están tocando entre sí?

Discute:
Permita que los estudiantes hagan una lluvia de ideas sobre cómo la computadora podría determinar si los dos sprites se están tocando. Haga una lista de sus ideas en el pizarrón y dígales que tendrán la oportunidad de probar sus teorías en un momento.

Meta: El propósito de esta discusión es solo para obtener algunas ideas en la pizarra para que los estudiantes las utilicen en la próxima Lección. No es necesario evaluarlos o probarlos, porque los estudiantes trabajarán juntos para hacerlo inmediatamente después de la discusión

Ampliación del conocimiento

Colisiones desconectadas

Grupo:
Agrupa a los estudiantes en parejas.

Observaciones
Ahora va a tener la oportunidad de probar las estrategias que surgieron como grupo. Cada guía de actividades tiene cuatro hojas de papel. Un compañero debe tomar los papeles con la “A” en la parte superior, y el otro debe tomar los papeles con la “B” en la parte superior. Cada uno dibujará dos sprites secretos en el gráfico, y su compañero tratará de averiguar si están tocando o no según la misma información que la computadora tendrá sobre las propiedades de cada uno de los sprites, así que no permita que su compañero mire lo que estás dibujando.

Distribuir:
La Detección de Colisión (Versión A) y Detección de Colisión (Versión B) para cada pareja. Asegúrese de que un compañero haya tomado las páginas con “A” en la parte superior y el otro haya tomado las páginas con “B” en la parte superior.
Cada alumno tendrá una línea para dibujar dos cuadrados. El estudiante elige la ubicación y el tamaño de cada uno de los cuadrados, y luego registra la información sobre los cuadrados en una tabla. A continuación, cambian las tablas (no los dibujos) y tratan de determinar si los dos sprites se están tocando en función del ancho de cada sprite y la distancia entre ellos.
La matemática para determinar si los sprites se están tocando es la siguiente:

- Resta las posiciones x (o y) de los sprites para encontrar la distancia entre sus centros.
- Divide el ancho (o la altura) de cada cuadrado por 2 para obtener la distancia desde el centro hasta el borde.
- Si la distancia entre los centros de los sprites es mayor que la suma de las distancias desde sus centros hasta sus bordes, los sprites no se tocan.
- Si la distancia entre los centros de los sprites es igual a la suma de las distancias desde sus centros hasta sus bordes, los sprites apenas se tocan.
- Si la distancia entre los centros de los sprites es mayor que la suma de las distancias desde sus centros hasta sus bordes, los sprites se superponen.

Recorra la sala:
Apoye a los estudiantes mientras completan la hoja de trabajo. Si los estudiantes no están seguros de cómo determinar si los sprites se están tocando, aliéntelos a usar una de las ideas en el pizarrón. Recuerde que no están siendo calificados sobre si tienen razón o no, sino sobre su capacidad para usar el proceso de resolución de problemas. Si alguno de los estudiantes termina antes de tiempo, guíelos para que encuentren un método que funcione para sprites en cualquier lugar de la cuadrícula, no solo en la misma línea.

Compartir:
Después de que todos los estudiantes hayan tenido la oportunidad de probar sus soluciones, pida que compartan lo que descubrieron.

Meta: El objetivo de esta discusiónes que los estudiantes vean que las matemáticas en el código son las mismas matemáticas descritas en la Lección desenchufada. El códigocombina las pruebas para x - y en declaraciones "si" anidadas.

Observaciones
La gente puede usar muchas estrategias diferentes para resolver un problema como éste. Debido a que las computadoras no pueden "ver" los dibujos de la misma manera que las personas, necesitan usar las matemáticas para determinar si dos cosas se están tocando. Veamos cómo esto puede funcionar a lo largo de una línea, pero podemos combinar estos métodos para trabajar en cualquier parte de la pantalla del juego.

IsTouching()

Transición:
Haga que la clase inicie sesión en la Unidad de Code Studio
Pantalla:
Muestre la burbuja 2 y permita que los estudiantes observen y analicen por parejas cómo podrían codificar el comportamiento que ven en el programa.

Niveles de Code Studio

El propósito de este nivel es que los estudiantes vean que pueden probar si dos sprites se están tocando con los bloques que ya están disponibles para ellos. El código es complicado, pero solo usa bloques que los estudiantes ya han aprendido a detectar cuando dos sprites se están tocando.

Discuta:
Pregunta a los estudiantes cómo se relaciona el código a continuación con lo que hicieron en la lección desconectada.
Observaciones
Aunque este código funciona, puede ser difícil de leer, y sería fácil cometer un error al escribirlo. También tomaría algo de tiempo escribir cada vez, por lo que un programador guardó el código en un bloque que podemos usar para determinar si dos sprites se están tocando, y no necesitamos escribir todo este código. Debido a que esto es algo que un programador quiere hacer una y otra vez, alguien ha creado un bloque llamado isTouching()que ya tiene todas las matemáticas dentro de él. Ocultar los detalles de cómo se hace algo para facilitar la programación se llama abstracción.

IsTouching()

Escribir las matemáticas cada vez que desees comprobar si dos sprites se están tocando puede llevar un tiempo, por lo que un programador creó el bloque isTouching, que puede comprobar si un sprite está tocando otro sprite (el objetivo). La computadora sigue haciendo los mismos cálculos que en el programa anterior, pero no tiene que preocuparse porque otro programador ya hizo ese trabajo.

Recorra la sala:
Apoye a los estudiantes mientras trabajan en los niveles 3-8. El nivel 8 permite a los estudiantes trabajar en el desplazamiento lateral que crearon en la última lección. Es muy abierto, así que anima a los estudiantes a ser creativos y tratar de codificar las interacciones de los sprites que no han visto antes. Los estudiantes que completan el nivel temprano pueden continuar agregando nuevas interacciones o mejorando sus juegos de otras maneras.

Transferencia del conocimiento (5 min)

Preguntar:
¿Cuáles fueron algunas de las cosas diferentes que tus sprites hicieron cuando interactuaron?
¿Cuál es un tipo de interacción que te gustaría, pero que aún no has visto?
¿Tienes alguna idea de cómo podrías escribir el código para ese tipo de interacción?

Compartir:
Permita a los estudiantes compartir el diferente tipo de interacciones que les gustaría ver. Hágales saber que aprenderán otros métodos para que los sprites interactúen más adelante.

Sugerencia para evaluación

Se sugiere los siguientes indicadores para evaluar formativamente los aprendizajes:

- Construyen y evalúan estrategias de manera colaborativa al resolver problemas no rutinarios.
- Incorporan el código, los medios y las bibliotecas existentes en los programas originales, para darles atribución.
- Prueban y perfeccionan sistemáticamente los programas.
- Documentan aquellos programas para que sean más fáciles de seguir, probar y depurar.

Lección 21: Movimiento complejo de Sprite

Lección en línea Ver en Code Studio

Propósito

Los estudiantes aprenden a combinar las propiedades de velocidad de los sprites con el patrón de contador para crear movimientos de sprites más complejos. En particular, los estudiantes aprenderán cómo simular la gravedad, hacer un sprite y permitir que un sprite flote hacia la izquierda o hacia la derecha. En los niveles finales de Code Studio, los estudiantes combinan estos movimientos para animar y controlar un solo sprite y crear un juego simple en el que un personaje vuela y recoge una moneda. Se anima a los estudiantes a hacer sus propias adiciones al juego en el nivel final.
Esta lección muestra la combinación de herramientas, en particular las abstracciones aprendidas en lecciones anteriores, les permite construir nuevos comportamientos para sus sprites. Esto resalta el punto más amplio de que las abstracciones no sólo simplifican el código, sino que también pueden usarse como componentes básicos de un comportamiento aún más complejo. La Lección presenta algunas de las programaciones más desafiantes. Los estudiantes combinarán múltiples construcciones de programación, incluyendo las propiedades de velocidad, el patrón del contador, la interacción del usuario y la detección de colisiones. Los patrones que los estudiantes usan en esta Lección generalmente son útiles para construir juegos y los estudiantes pueden reutilizarlos más adelante.

Secuencia para el aprendizaje

Conocimiento inicial (10 min)
Ampliación del conocimiento (60 min)
Transferencia del conocimiento (10 min)

Objetivos

Los estudiantes serán capaces de:

- Usar velocidad de sprite con el patrón de contador para crear diferentes tipos de movimiento de sprite.
- Explicar cómo las construcciones de programación individual pueden combinarse para crear un comportamiento más complejo.

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos
- Velocidad y patrón del contador – Recurso

Estrategia de aprendizaje

Conocimiento inicial (10 min)

Comentario:
Los cuatro conceptos principales que los estudiantes aprenderán en la Lección de hoy son isTouching(), keyDown(), sprite.velocityX/ sprite.velocityY. Pida a los estudiantes que hablen con un compañero y recuerden mutuamente para qué se usa cada una de estas cuatro construcciones y cómo funcionan. Como clase, defina cada uno según lo conversado.

Observaciones
En las últimas actividades aprendimos mucho de construcciones de programación poderosas que nos permitieron crear juegos mucho más interesantes. Hoy vamos a explorar cómo la combinación de estas nos dará aún más control sobre los tipos de juegos que podemos hacer.

Ampliación del conocimiento (60 min)

Transición:
Dirija los estudiantes a Code Studio, Contenido de la Lección - code.org. Con la excepción de la discusión posterior al primer nivel, los estudiantes trabajarán en Game Lab hasta el final de la lección.

Resumen
La clase aprende a combinar las propiedades de velocidad de los sprites con el patrón de contador para crear un movimiento de sprite más complejo, como simular la gravedad, hacer que un sprite salte y permitir que un sprite flote hacia la izquierda o hacia la derecha. En los niveles finales, la clase combina estos movimientos para animar y controlar un solo sprite y construir un juego simple en el que un personaje vuela y recoge monedas.

Este nivel introduce el nuevo patrón de programación principal de esta Lección, combinando el patrón de contador conlas propiedades de velocidad de los sprites. Anime a los estudiantes a tomaren serio sus predicciones antes de ejecutar realmente el código.

Discute:
Después de que los estudiantes hayan hecho predicciones sobre cómo se ejecutó rápidamente el código en el primer nivel, analicen por qué vieron que el automóvil comenzó a moverse más rápido. Ten en cuenta que mientras que antes usaban el patrón de contador para aumentar la posición de un automóvil, ahora se usa para aumentar la velocidad del automóvil.

Niveles 2-4:
Estos niveles les dan a los estudiantes la práctica de usar la velocidad dentro del patrón del contador.

Velocidad y el patrón contador

Usar una sprite.velocityX propiedad con el patrón de contador cambiará la velocidad de un sprite durante el programa. Esto hace que el sprite se acelere. Practica un poco usando este patrón tú mismo.

Niveles 5 - 6:
Estos niveles introducen el uso de la velocidad y el patrón del contador para ralentizar un sprite, eventualmente moviéndolo en la dirección opuesta. Esto lleva a la introducción de cómo este patrón podría usarse para simular la gravedad.

Niveles 7 a 9:
Los estudiantes comienzan a trabajar en un juego de volante. En estos niveles usan las construcciones de programación que han aprendido para hacer que su personaje principal se mueva. El personaje responde a gravedad simulada, saltos y flotantes a izquierda y derecha.

Saltando

Aumentar la velocidad de un sprite dentro del patrón de contador puede simular la gravedad. Al agregar interacciones del usuario, puedes hacer que tu sprite parezca que también salte.

Flotante a la derecha

Ahora están usando el patrón de contador con la velocidad Y del sprite para simular la gravedad y saltar. Si usan la velocidad X del sprite en el patrón contrario, pueden hacer que tu sprite flote de lado a lado también.

Niveles 10 a 12:

Los estudiantes agregan una moneda al juego para que su personaje la coleccione. En el último nivel, se les anima a actualizar el juego ellos mismos. Aprovecha esta oportunidad en particular para alentar a los estudiantes a usar otros patrones de programación que hayan aprendido, por ejemplo, crear un marcador.

Transferencia del conocimiento (10 min)

Compartir:
Pida a los estudiantes que compartan con sus compañeros de clase qué adiciones hicieron a su último juego. Haga que los estudiantes se centren no sólo en cómo funciona el juego, sino también en cómo se ve el código para crear ese tipo de funcionalidad.
Preguntar:
En su papel, hagan dos listas. Primero una lista de cosas nuevas que puedan programar después de la Lección de hoy. En la segunda lista anoten todos los nuevos bloques que aprendieron hoy.
Discute:
Haga que los estudiantes compartan sus listas con sus compañeros de clase. Después comparte listas como una clase. Deberían haber enumerado muchos nuevos movimientos de sprites pero los estudiantes no han aprendido ningún bloque nuevo en esta Lección.
Después de avisar que todos los movimientos nuevos que crearon hoy se hicieron combinando bloques y patrones que ya aprendieron, pregunte a los estudiantes qué piensan, qué les puede decir esto sobre cómo los programadores desarrollan el código.
Indicación:
Hoy hemos construido muchos movimientos de sprites nuevos, como la gravedad y el salto, pero nada de esto nos obligó a aprender nuevos bloques. ¿Crees que aprender a programar siempre significa aprender nuevos comandos?
Discute:
Lidere un seguimiento rápido de su discusión inicial sobre este punto.
Observaciones
Vamos a seguir aprendiendo algunas herramientas más en Game Lab. Para crear nuevos tipos de programas, no siempre es necesario aprender nuevos bloques. La mayoría de las veces, la creatividad de la programación proviene de aprender a combinar cosas que ya conoces de maneras nuevas y creativas.

Objetivo: Esta conversación debe resaltar que los estudiantes no aprendieron ningún bloque nuevo en la Lección de hoy, sino que aprendieron nuevas formas de combinar bloques y patrones que habían aprendido previamente. El punto más amplio aquí es que la programación no siempre se trata de aprender nuevos bloques, sino de ser creativos al combinar las herramientas que ya sabes usar en el lenguaje.

Meta: Reforzar el hecho de queaprender a programar no es solo memorizar bloques. Ser creativo con laprogramación a menudo significa idearmaneras inteligentes de combinar los comandos y patrones que ya sabe usar.

Sugerencias para evaluación

Se sugieren los siguientes indicadores para evaluar formativamente los aprendizajes:

- Crean nuevos tipos de programas, combinando elementos que ya conocen de manera nueva y creativa.
- Perfeccionan el trabajo involucrando modificaciones y testeo.

Lección 22: Colisiones

Lección en línea ver en Code Studio

Propósito

Los estudiantes programan sus sprites para interactuar de nuevas maneras. Después de una breve revisión de cómo usaron el bloque isTouching, los estudiantes intercambian ideas sobre otras formas en que dos sprites podrían interactuar. A continuación, utilizan isTouching para hacer que un sprite empuje al otro a través de la pantalla antes de practicar con los cuatro bloques de colisión (collide, displace, bounce, y bounceOff).

Esta lección introduce colisiones, otra abstracción útil que permitirá a los estudiantes manipular sus sprites de maneras completamente nuevas. Si bien los estudiantes teóricamente pueden haber escrito sus propios comandos de desplazamiento, colisión o rebote, la capacidad de ignorar los detalles de este código les permite centrar su atención en la estructura de alto nivel de los juegos que desean construir.

Esta Lección también pretende que los estudiantes practiquen utilizando los nuevos comandos que han aprendido. En realidad, es la última vez que aprenderán un nuevo comportamiento de sprite, y después de esta lección, los estudiantes pasarán a centrarse más en cómo organizan su código cada vez más complejo.

Secuencia para el aprendizaje

Conocimiento inicial (5 min)
Ampliación del conocimiento (40 min)
Transferencia del conocimiento (10 min)

Objetivos

Los estudiantes serán capaces de:

- Utilizar los bloques “displace”, “collide”, “rebunce”, y “bounceOff” para producir interacciones de sprites
- Describir cómo se pueden construir las abstracciones para desarrollar aún más abstracciones.
- Para los profesores:
- CSD Unidad 3 - Animaciones y Juegos Interactivos

Vocabulario

Abstracción: una representación simplificada de algo más complejo. Las abstracciones le permiten ocultar detalles para ayudarlo a manejar la complejidad, enfocarse en conceptos relevantes y razonar sobre problemas en un nivel superior.

Código

- sprite.bounce(target)
- sprite.bounceOff(target)
- sprite.collide(target)
- sprite.displace(target)
- setCollider(type, xOffset, yOffset, width/radius, height, rotationOffset)
- sprite.bounciness

Estrategia de aprendizaje

Conocimiento inicial (5 min)

Pantalla:
Si tiene la capacidad, proyecte la animación en el primer nivel en Code Studio para esta lección. De lo contrario, pida a las parejas de estudiantes que lo vean juntos. Muestre dos sprites, uno moviéndose a través de la pantalla hacia el otro y finalmente empujando uno cuando colisionan.

Resumen
La clase programa sus sprites para interactuar de nuevas maneras. Después de una breve revisión de cómo usaron el bloque isTouching, la clase intercambia ideas sobre otras formas en que dos sprites podrían interactuar. Luego usan isTouching para hacer que un sprite presione otro en la pantalla, antes de practicar con los cuatro bloques de colisión (colisionar, desplazar, rebotar y rebotar).

Indicación:
Usando los bloques que ya sabemos cómo usar, ¿cómo podríamos crear la interacción de sprites que podemos ver en este programa?

Meta: El objetivo de esta discusión es que los estudiantes piensen en formas de resolver el problema de tener un elemento sprite empujando otro enla pantalla. No es necesario que los estudiantes lleguen a un consenso, ya que cada uno de ellos tendrá la oportunidad de probar una solución en el siguiente nivel en Code Studio. Los estudiantes deben entender quees posible usar bloques para producir el movimiento deseado solo con los bloques que ya han aprendido.

Observaciones
Tenemos muchas ideas geniales sobre cómo podríamos hacer que un sprite empuje otro en la pantalla. Ahora que ya te has preparado, puedes probar tus ideas en Code Studio. Una gran parte del problema es averiguar cuándo se tocan los dos sprites, pero como ya hemos descubierto cómo hacerlo y ahora podemos usar el bloque isTouching, ya no necesitamos pensarlo. Podemos enfocarnos en la nueva parte del problema.

Ampliación del conocimiento

Niveles de Code Studio

Niveles 2-4:
En estos niveles, se muestra a los estudiantes una interacción de sprites. Luego implementan sus ideas para crear la interacción de los sprites que observaron. La primera vez pueden implementar las ideas del grupo. La segunda vez desafíalos a implementar ese comportamiento de forma independiente.

Interacciones de Sprite

Hasta ahora han sido capaz de crear interacciones de sprites simples usando el sprite. bloque isTouching(). Por ejemplo, han reestablecido una moneda a una ubicación diferente en la pantalla cuando un personaje la toca. Ahora es el momento de comenzar a hacer que los sprites tengan interacciones más complejas.

Hacer esto

- Ejecuta el programa y observa la interacción entre los dos sprites.
- Discute con un compañero: usando solo los comandos que ya conoces, ¿cómo podrías crear este tipo de interacción? Hay muchas maneras de hacerlo, pero aquí hay algunos bloques a considerar:
    - sprite.isTouching()
    - sprite.velocityX
    - sprite.velocityY
    - sprite.x
    - sprite.y

Prepárate para compartir tus ideas con tus compañeros de clase.

Inducción:
Éste fue un problema desafiante, pero pudimos resolverlo.
¿Qué nos ayudó a resolver este problema?

Observaciones
Todas estas cosas son muy importantes y surgen mucho en Informática. Una cosa que fue particularmente útil fue el isTouching bloque, que ocultó el código complicado que nos dice si los dos sprites se están tocando. También hay un bloque que oculta el código que acabamos de escribir, y algunos otros bloques que ocultan el código para otros tipos de interacciones de sprites. Tendrás la oportunidad de probar estos bloques en los siguientes niveles y utilizarlos para mejorar tu juego de vuelo.

Niveles 5-10:
Estos niveles introducen cómo usar los 4 nuevos bloques de colisión que los estudiantes aprenderán en esta lección.

Desplazar
sprite.displace() hará que un sprite empuje al otro cuando se toquen.

Tipos de colisión
Hay cuatro tipos de colisiones que usamos en Game Lab. Estos bloques causarán un cierto tipo de interacción entre el sprite y su objetivo.

Desplazar
El bloque displace hace que el sprite empuje al objetivo siempre que se toquen entre sí.
El sprite se mueve normalmente.

Chocar
El bloque collide hace que el sprite se detenga cuando se encuentra con el objetivo. Si el objetivo se está moviendo, empujará al objeto con él. El objetivo sigue moviéndose normalmente.

Rebotar
El bloque bounce hace que el sprite y el objetivo reboten cuando se tocan entre sí. Tanto el sprite como el objetivo cambian cómo se están moviendo. El bloque bounceOff hace que el sprite rebote en el objetivo. El objetivo sigue moviéndose normalmente.

Depurar
Hay una sprite.debug propiedad especial que puede usar para comprender mejor por qué los sprites interactúan de la manera en que lo hacen.

Este es un buen momento para decir cuánto han progresado los estudiantes en sus habilidades desde el comienzo de la unidad. Este problema habría parecido casi imposible a principios deaño. Algunas cosas que hicieron que elproblema fuera más fácil de resolver fueron:

- Preparación: Los estudiantes intercambiaron ideas y pensaron en soluciones antes de probar sucódgo
- Cooperación: Los estudiantes trabajaron en grupos para llegar auna solución
- Abstracción: los estudiantes pudieron usar los bloques **isTouching** y **velocityY** para ocultarparte de la complejidad de la solución

SetCollider
Los Sprites interactúan en función del tamaño y la forma de su colisionador, no de las imágenes que se les asignan. Solo se puede ver el colisionador cuando se activa el modo de depuración. Puede cambiar la forma del colisionador usando el sprite.setCollider() bloque, que le permite elegir entre un “rectángulo” o un “círculo”. Por defecto, todos los colisionadores son “rectangulares”.
Niveles 11-15:
Estos niveles guían a los estudiantes al agregar colisiones al juego que comenzaron en la Lección anterior, y eventualmente invitan a los estudiantes a agregar sus propias modificaciones al juego.

Bounciness
Hasta el momento, bounceOff ha hecho que los sprites se alejen de otros objetos tan rápido como rebotaban en ellos. En el mundo real, casi todo se ralentiza un poco cuando rebota en otra cosa. Puedes usar el bounciness bloque para decirle a tu sprite cuánto desacelerar o acelerar cuando rebota en otra cosa.

Transferencia del conocimiento (10 min)

Compartir y publicar 3-2-1
Permita a los estudiantes tiempo para jugar los juegos de los demás. Pida que se centren no sólo en el nuevo comportamiento que agregaron, sino también en el código que usaron para crearlo.
Revise:
Pida a los estudiantes que escriban y reflexionen sobre las siguientes indicaciones.

- ¿Tres cosas que viste en el juego de otra persona que realmente te gustaron?
- ¿Cuáles son dos mejoras que harías en tu juego si tuvieras más tiempo?
- ¿Cuál es un bloque que te gustaría tener en Game Lab y cómo funcionaría?

Sugerencias para evaluación
Se sugiere el siguiente indicador para evaluar formativamente los aprendizajes:

- Utilizan bloques para programar en base un diseño predictivo
- Explican la forma en que diseñan su programa.

Lección 23: Funciones

Lección en línea Ver en Code Studio

Propósito

Los estudiantes aprenden cómo crear funciones para organizar su código, hacerlo más legible y eliminar bloques repetidos de código. Una actividad desconectada de calentamiento explora cómo las direcciones en diferentes niveles de detalle pueden ser útiles dependiendo del contexto. Los estudiantes aprenden que los pasos de mayor nivel o más abstractos facilitan la comprensión y razonamiento de los pasos. Luego, los estudiantes aprenden a crear funciones en Game Lab. Usarán funciones para eliminar bloques largos de código y para reemplazar piezas repetidas de código con una sola función. Al final de la Lección, los estudiantes usan estas habilidades para organizar y agregar funcionalidad a la versión final de su juego de desplazamiento lateral.

En las primeras cuatro lecciones, los estudiantes han aprendido a utilizar una serie de abstracciones en sus programas, que incluyen las propiedades de velocidad, el toque y las colisiones. Estas abstracciones les han permitido crear programas mucho más complejos sin tener en cuenta los detalles de cómo se crea ese comportamiento. En esta Lección, los estudiantes aprenden a crear abstracciones propias mediante la creación de funciones. Los estudiantes utilizarán principalmente las funciones para dividir el código en fragmentos lógicos que son más fáciles de razonar. Esto forma parte de la transición de la construcción de habilidades técnicas a los procesos organizacionales utilizados para desarrollar software.

Secuencia para el aprendizaje

Conocimiento inicial (10 min)
Ampliación del conocimiento (60 min)
Transferencia del conocimiento (10 min)

Objetivos

Los estudiantes serán capaces de:

- Crear y usar funciones para bloques de código que realizan una sola tarea de alto nivel dentro de un programa
- Crear y usar funciones para eliminar bloques repetidos de código de sus programas
- Crear y usar funciones para mejorar la legibilidad de sus programas
- Explicar cómo las abstracciones permiten a los programadores razonar sobre un programa en un nivel superior.
- Para los profesores:
- CSD Unidad 3 - Animaciones y Juegos Interactivos
- Funciones - Recurso
- Para los estudiantes:
- Funciones – Video

Vocabulario

- **Función:** un grupo nombrado de instrucciones de programación. Las funciones son abstracciones reutilizables que reducen la complejidad de escribir y mantener programas.

Código

- Define a function
- Call a function

Estrategia de aprendizaje

Conocimiento inicial (10 min)

Instrucciones de alto nivel vs. Nivel bajo

Preguntar:
Imagina que necesitas escribir en un conjunto de instrucciones de 5 pasos para pasar la mañana. ¿Qué serían? Escribe sus pasos y prepárese para compartir.

Discute:
El alumno debe escribir las instrucciones de 5 pasos para la mañana y compartir con un compañero. pide a un par de estudiantes que compartan con la clase.

Inducción:
Este conjunto de instrucciones es bastante fácil de seguir y comprender. Están en el nivel en el que podrías pensar en describir tu día a un amigo. Ahora vamos a un nivel más profundo. Elija uno de tus 5 pasos y divídelo en 5 pasos más pequeños que necesitas para completar esa tarea más grande. Prepárese para compartir sus ideas de nuevo.

Discute:
Los estudiantes deben compartir sus pasos más pequeños. Nuevamente, solicita a algunos voluntarios que luego compartan su paso original y cómo lo separaron.
Esto se está poniendo interesante. Parece que la primera vez que dimos nuestros pasos, estábamos "ocultando" algunos de los detalles necesarios para completar la tarea.
Probemos esto una vez más. Toma uno de sus 5 pasos más pequeños y divídelo de nuevo en 5 pasos aún más pequeños.
Pida a los estudiantes que compartan sus pasos una vez más. Pide nuevamente a algunos voluntarios que compartan cómo dividieron uno de sus pasos en pasos aún más pequeños.

Indicación:
Imagina que dividimos cada uno de los primeros 5 pasos en 5, y luego dividimos todos esos pasos nuevamente. Esto significa que tendríamos un conjunto de instrucciones de alto nivel, y en la parte inferior un conjunto de instrucciones realmente bajo o detallado.
Prepárate para responder a las siguientes preguntas.

- ¿Cuándo sería más apropiado el conjunto de pasos de alto nivel que acabas de escribir?
- ¿Cuándo sería más apropiado el conjunto de pasos de nivel más bajo?
- ¿Cuál es más fácil de razonar o entender?
  Discutir:

Pida a los estudiantes que compartan sus pensamientos y opiniones. Después de un par de minutos cambie la conversación al comentario de transición a continuación.

Observaciones
A veces los detalles son importantes, pero a menudo los pasos de alto nivel son mucho más fáciles de razonar y dejan claro lo que está sucediendo. Hemos aprendido que bloques como velocityX o isTouching en realidad solo contienen código que podríamos haber escrito nosotros mismos. Usar estos comandos, o abstracciones, es realmente útil ya que podemos pensar en el código a un alto nivel. Hoy aprenderemos cómo agrupar muchos comandos para crear un bloque nuevo. En la programación cuando creamos un nuevo bloque como este, lo llamamos función.

Objetivo: Esta conversación demuestra a los estudiantes que a menudo usan un nombre o descripción de alto nivel para un comportamiento mucho más complejo. Está motivando el valor de agrupar o combinar muchos pasos más pequeños con un nombre más grande y proporciona alguna de las justificaciones para crear funciones dentro de los programas. A saber, los pasos de alto nivel hacen que sea másfácil de entender y razonar sobre un proceso.

Proceso de desglose: El orden en el que están desglosando una tarea más grande aquí también reflejacómo se les pedirá que escriban código con funciones. A menudo, los programadores primero escriben el nombre de la función que pretenden escribir en función de lo que debede hacer antes de entrar y escribir los detalles.

Ampliación del conocimiento (60 min)

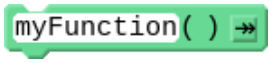
Transición:
Dirija a los estudiantes a Code Studio, contenido de la Lección donde aprenderán a crear funciones.
Resumen
Esta lección cubre las funciones como una forma de organizar su código, hacerlo más legible y eliminar bloques repetidos de código. La clase aprende que los pasos de mayor nivel o más abstractos facilitan la comprensión y razonamiento de los pasos, luego comienza a crear funciones en Game Lab. Al final de la lección, la clase usa estas habilidades para organizar y agregar funcionalidad a la versión final de su juego de desplazamiento lateral.

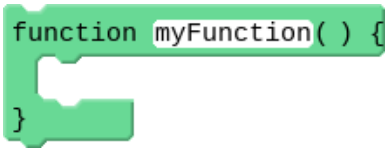
Vocabulario

- **Función**: una pieza de código que se puede llamar fácilmente una y otra vez.

Funciones de llamada

Las funciones permiten construir tus propios bloques y decidir qué código va dentro de ellos. Este es el comando que le permite crear una nueva función.
Usas o "llamas" tu función como cualquier otro bloque usando el nombre que le diste.
Los bloques para crear y llamar a funciones se pueden encontrar en la pestaña "Funciones" de la paleta.





Reordenando el código

Colocar el código dentro de las funciones facilita la lectura y la realización de cambios. Puedes llamar a tus funciones en un orden diferente para realizar rápidamente cambios significativos en la ejecución de tu programa.

Funciones de llamada en Ciclos de Dibujo

Puede llamar a una función dentro del ciclo de extracción, tal como lo haría en cualquier otro lugar de su código.

Llamar a las funciones varias veces

A menudo querrá usar el mismo código en muchos lugares de su programa. Una vez que haya creado una función, puede llamarla tantas veces como desee.

Hacer cambios en las funciones

Una buena ventaja de utilizar funciones para eliminar código repetido es que ahora puedes realizar fácilmente cambios en múltiples lugares en su código. Simplemente cambia la forma en que crea las funciones, y su programa ahora usará el nuevo código en todas partes donde se llame tu función.

Introduciendo funciones
En estas primeras lecciones a los estudiantes simplemente se les muestra la sintaxis de las funciones y no se les pide que escriban o creen las suyas propias. Puede ser útil explicar la creación de una función como básicamente "crea un nuevo bloque" al igual que otro programador creó losbloques "isTouching" o "velocity" que han visto realmente contienen otro código más complejo.

Por qué usar las funciones
Este y los siguientes dos nivelespresentan tres usos de funciones, asaber, eliminar la repetición en los programas, permitir que el código se cambie rápidamente en varios puntos yproporcionar organización en el código. Los estudiantes necesitarán escribir más de sus propias funciones en estos niveles.

Meta: Use este primer mensaje para realizar lo que los estudiantes aprendieron hoy. Cuando crean una función, crean su propio bloque que pueden llamar o usar cuando lo deseen. Vieron al menos dos motivaciones principales para crear funciones hoy en día, incluidas.

- Simplificando el código dividiendoen fragmentos lógicamente nombrados
- Evitando el código repetido haciendo un bloque, puedes usarlovarias veces

Transferencia del conocimiento (10 min)

Pregunte:
¿Por qué diríamos que las funciones nos permiten "crear nuestros propios bloques"?
¿Por qué es esto algo que quisiéramos hacer?
Discute: Haz que los estudiantes discutan en su mesa antes de hablar como clase.
Preguntar: Escribe tu propia definición de abstracción ¿Por qué una función cuenta como una abstracción?
Discute: Haz que los estudiantes discutan en su mesa antes de hablar como clase.
Observaciones
Las funciones son una herramienta útil para ayudarnos a escribir y
Preguntar: Escribe tu propia definición de abstracción ¿Por qué una función cuenta como una abstracción?
Discute: Haz que los estudiantes discutan en su mesa antes de hablar como clase.

Observaciones

Las funciones son una herramienta útil para ayudarnos a escribir y organizar fragmentos de código más complejos. Poder mantener tu código organizado será una habilidad importante.

Sugerencias para evaluación

Se sugiere el siguiente indicador para evaluar formativamente los aprendizajes:

- Agrupan la información en unidades más pequeñas
- Explican cómo las funciones hacen el trabajo más eficiente

Lección 24: El proceso de diseño de juego

Lección en línea Ver en Code Studio

Propósito

Esta lección presenta a los estudiantes el proceso que usarán para diseñar juegos. Este proceso se centra en una guía de proyecto que les pide a los estudiantes que definan sus sprites, variables y funciones antes de comenzar a programar su juego. En esta Lección, los estudiantes comienzan jugando un juego en Game Lab donde el código está oculto. Discutan qué creen que deberían hacer los sprites, las variables y las funciones para hacer el juego. Luego reciben una guía de proyecto completa que muestra una forma de implementar el juego. A continuación, los estudiantes pasan por este proceso a través de una serie de niveles. Como parte de esta Lección, los estudiantes también aprenden brevemente a utilizar animaciones de cuadros múltiples en Game Lab. Al final de la Lección, los estudiantes tienen la oportunidad de hacer mejoras en el juego para hacerlo suyo.

Esta Lección presenta animaciones de cuadros múltiples y es la primera de una secuencia centrada en el proceso de creación de software.

Si bien las lecciones anteriores se centraron en el uso de abstracciones para gestionar la complejidad del código, esta lección se centra en la gestión de la complejidad del proceso de desarrollo de software. La lección ancla en gran medida el proceso de desarrollo de software al proporcionar a los estudiantes una guía de proyecto completa, proporcionando código de inicio, y encaminando a los estudiantes a través de su implementación. En las lecciones posteriores, los estudiantes deberán completar una mayor parte de esta guía de forma independiente, y para el proyecto final, seguirán este proceso.

Secuencia para el aprendizaje

Conocimiento inicial (15 min)
Ampliación del conocimiento (60 min)
Transferencia del conocimiento (20 min)

Objetivos

Los estudiantes serán capaces de:

- Identificar las construcciones de programación central necesarias para construir diferentes componentes de un juego.
- Crear y usar animaciones de multitrama en un programa.
- Implementar diferentes características de un programa siguiendo una guía de proyecto estructurado.

Preparación

- Contenido de la Lección - code.org

Recursos

¡Atención!
Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los Profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos

Para los estudiantes:

- Juego Defensor

Estrategia de aprendizaje

Conocimiento inicial (15 min)

Jugar Cake Defender

Transición:

Esta lección comienza inmediatamente en Code Studio Contenido de la Lección - code.org. En el primer nivel, encontrarán un juego, pero no podrán ver el código. Deben jugar el juego y seguir las instrucciones que les piden que enumeren las variables, los sprites y las funciones que creen que son necesarios para crear este juego.

Mantener el enfoque: los estudiantes pueden distraerse fácilmente con la diversión de jugar el juego. Déjelos jugar por un tiempo, pero eventualmente anímelos a seguir las instrucciones en pantalla y haga una lista de las variables, los sprites y las funciones que sería necesarias para jugar el juego.

Ejecución de la conversación: puede escribir "variables", "sprites" y "funciones" en la pizarra y registrar sus ideas debajo de cada una. Pídales a los estudiantes que justifiquen sus decisiones, pero no sientan la necesidad de conformarse con una sola respuesta correcta

¡Defiende tu Torta!

Es un ejemplo de un juego que construirá al final de esta Lección.

Detener: revisar la guía de proyectos

Debate:

Los estudiantes deberán crear individualmente una lista de variables, sprites y funciones que crean que se utilizan en el juego. Pida a los estudiantes que compartan sus listas con un compañero antes de hablar como clase.

Distribuir:

Entregue a cada alumno o pareja de estudiantes una copia del Juego Defender - Guía de proyectos

Preguntar:

Compara los componentes del juego que creías que se incluirían con los de esta guía de proyecto. ¿Notan alguna diferencia?

Discute:

Como clase, compara la lista que tenías en el pizarrón con la lista de variables, sprites y funciones en la guía del proyecto. Ten en cuenta las similitudes. Donde hay diferencias intenta entender por qué. Evita "correcto" o "incorrecto".

Observaciones

Por lo general, hay muchas maneras de estructurar un programa para que funcione de la manera que desees. Lo importante al escribir programas complejos o grandes es que comienzas con un plan. Hoy vamos a ver cómo podemos implementar este plan para construir nuestro propio juego. Al final de la Lección, no sólo habrás construido tu juego, sino que sabrás cómo cambiarlo y hacerlo tuyo. ¡Vamos!

Guía del proyecto: la guía del proyecto se completa de manera exhaustiva para los estudiantes, de modo que pueda experimentar su uso como referencia cuando programe. Esto debería darles más contexto al completar su propia guía de proyectos en las próximas dos lecciones.

Puede dar a cada uno su propia copia como referencia, pero también puede optar por imprimir una copia por par, compartir copias digitales o simplemente mostrar la guía en el proyector. Siempre que esté disponible para referencia, cualquier enfoque funcionará bien.

Ampliación del conocimiento (60 min)

Animaciones de multitrama

Transición:

Los estudiantes deben regresar a Code Studio. Antes de implementar realmente el plan, los estudiantes deberán revisar rápidamente una nueva habilidad, cómo usar animaciones de multitrama. Los estudiantes aprenderán rápidamente a crear, modificar y cambiar el nombre de las animaciones, así como a elegirlas de la biblioteca.

Detener

Antes de seguir adelante, deberá consultar la Guía del proyecto.

Implementar la guía del proyecto

Los estudiantes reciben una gran cantidad de código de inicio en este proyecto. Los sprites, las variables y las funciones ya se les han asignado. El trabajo de este proyecto es escribir el código para las funciones individuales. Estos niveles guían a los estudiantes a través de cómo implementar esas funciones. A medida que los estudiantes avanzan por los niveles, señalan cómo se está utilizando la guía del proyecto.

Las habilidades más desafiantes que usan los estudiantes en estos niveles es reconocer la necesidad de crear nuevas funciones para reemplazar el código repetido. Los estudiantes necesitan desarrollar esta habilidad por su cuenta, pero estos niveles demuestran una instancia donde esto podría suceder.

Realizar los niveles de Code Studio correspondiente a la Lección.

Transferencia del conocimiento (20 min)

Hazlo tuyo

Este último nivel anima a los estudiantes a hacer suyo el juego. Si los estudiantes han llegado a este punto, tienen todas las habilidades que necesitan para progresar a través del plan de estudios, por lo que no hay presión para completar ninguna de las modificaciones sugeridas en este nivel. Sin embargo, si tienen tiempo, obtener una práctica de planificación e implementar nuevas características será una habilidad útil. Incluso modificar las animaciones del juego es una forma sencilla de que los estudiantes puedan apropiarse del juego.

Compartir:

Una vez que los estudiantes han completado el proyecto, pueden compartir su trabajo con sus compañeros de clase. Anime a los estudiantes a exhibir el código adicional que escribieron y explique cómo ha cambiado la forma en que funciona el juego.

Sugerencias para evaluar

Se sugiere los siguientes indicadores para evaluar formativamente los aprendizajes:

- Elaboran una guia de programa
- Explican la finalidad de las diferentes estructuras utilizadas
- Colaboran de forma grupal en el desarrollo del diseño del juego

Lección 25: Uso del proceso de diseño de juego

Lección en línea Ver en Code Studio

Propósito

En esta lección de varios días, los estudiantes usan el proceso de resolución de problemas para crear un juego de salto de plataforma. Comienzan por mirar un ejemplo de un saltador de plataforma y luego definen cómo se verán sus juegos. A continuación, utilizan un proceso estructurado para planificar los fondos, las variables, los sprites y las funciones que necesitarán para implementar su juego. Después de escribir el código del juego, los estudiantes reflexionarán sobre cómo se puede mejorar el juego e implementarán esos cambios.
Los estudiantes ya han aprendido todos los constructos de programación que necesitan para hacer un juego. Esta lección repasa muchos de esos conceptos al tiempo que los presenta por medio de un proceso estructurado que los ayudará a administrar el trabajo. Se basa en el uso de la Guía del proyecto de la lección anterior, donde los estudiantes la completan aún más de forma independiente, antes de usarla para construir un juego. Esta lección prepara a los estudiantes a escribir su propio juego desde cero para el proyecto final.

Secuencia para el aprendizaje

Conocimiento inicial (10 min)
Ampliación del conocimiento (50 min)
Transferencia del conocimiento (5 min)

Objetivos

Los estudiantes serán capaces de:

- Identificar las construcciones de programación central necesarias para construir diferentes componentes de un juego.
- Implementar diferentes características de un programa siguiendo una guía de proyecto estructurado.

Preparación

- Planificación de su plataforma de juego

Recursos

> ¡Atención!
> Por favor, haga una copia de cada documento que planee compartir con los estudiantes.

Para los profesores:

- CSD Unidad 3 - Animaciones y Juegos Interactivos
- Para los estudiantes:
- Planificando tu Juego de Plataforma

Estrategia de aprendizaje

Conocimiento inicial (10 min)

Introducción

El proceso de resolución de problemas nos ayuda a resolver todo tipo de problemas. Piensa en el problema de construir una pieza de software más grande, como el juego que construimos en la última lección. ¿Cómo se veía cada uno de los 4 pasos? ¿Por qué fueron importantes?

Discute:

Los estudiantes deben intercambiar ideas en voz baja y anotar cada paso. Después, dirija una discusión de compartir. Puede grabar ideas en el tablero. Las posibles fases incluyen:

- Definir: Averiguar cómo quieres que se vea el juego, cómo debería funcionar y quién lo jugará.
- Preparar: Planificar cómo se verá tu código. Decide una estructura para tu juego.
- Intentar: Escribir el código que sigue a su plan.
- Reflexiona: probar su código, jugar el juego para asegurarse de que funciona, recibir comentarios de otras personas para mejorar el juego.

Meta: Los estudiantes deben compartir sus pensamientos, pero si no surgen naturalmente, sugiérales los ejemplos proporcionados. Esta discusión motivará el uso de la guía del proyecto para construir un juego más adelante en la lección.

Observaciones

Cuando se construye un software, el proceso de resolución de problemas puede ser una guía útil. Obviamente, necesitamos escribir el código, pero tener cuidado de definir lo que queremos construir, hacer un buen plan para construirlo y reflexionar luego sobre cómo mejorarlo son parte de la creación de un buen software. Hoy vamos a usar este proceso para hacer un nuevo juego.

Ampliación del conocimiento

Jugar Alien Jumper

Distribuir:

Entregue a cada alumno una copia de la planificación de su **Guía de plataforma de juego**

Observaciones

Vamos a construir un juego de saltos. Tendrá la oportunidad de jugar un juego de muestra y luego planificar cómo crear el juego en su Guía de proyectos.

Dirija a los estudiantes a Code Studio. En el primer nivel, encontrarán un juego, pero no podrán ver el código. Deben jugar el juego y seguir las instrucciones que les piden, que enumeren las variables, los sprites y las funciones que creen que son necesarios para crear este juego.

Discutir la guía de proyectos

Recorra la sala:

Los estudiantes deben completar la guía del proyecto en el estilo de la que vieron en la lección anterior. Es probable que quieran mantener el juego mientras intentan determinar el comportamiento que tendrán cada uno de los sprites.

Compartir:

Los estudiantes comparten sus planes para hacer el juego. Reafírmeles que hay muchas maneras correctas de crear la misma pieza de software, y que tendrán la oportunidad de probar sus ideas en Code Studio.

- Los estudiantes trabajan en parejas para crear el juego.
- Los estudiantes tienen la oportunidad de mejorar su juego después de estar expuestos a otras dos versiones de un saltador de plataforma.
- Los estudiantes comparten sus juegos con sus compañeros de clase.

Hacer el juego tomará al menos dos periodos de clase. Si no hay suficiente tiempo para que todos los estudiantes terminen la clase, los grupos de estudiantes pueden trabajar para codificar diferentes aspectos y luego compartir el código entre ellos. Por ejemplo, un grupo podría trabajar en las plataformas, uno en las estrellas y otro en el jugador. Los estudiantes que terminaron temprano pueden elegir más desafíos de los niveles posteriores.

Reflexión

Preguntar:

Pida a los estudiantes que reflexionen sobre el desarrollo de las cinco prácticas de Descubrimientos CS (Resolución de problemas, Persistencia, Creatividad, Colaboración, Comunicación). Elija una de las siguientes indicaciones según lo considere apropiado.

- Elija una de las cinco prácticas en las que cree que demostró crecimiento en esta Lección. Escribe algo que hiciste que ejemplifica esta práctica.
- Elija una práctica en la que piense que puede seguir creciendo. ¿Qué le gustaría mejorar?
- Elija una práctica que pensó que era especialmente importante para la Lección que completamos hoy. ¿Qué lo hizo tan importante?

Sugerencias para evaluación

Se sugiere los siguientes indicadores para evaluar formativamente los aprendizajes:

- Implementan un diseño siguiendo la guía del proyecto
- Explican la finalidad de las diferentes estructuras utilizadas
- Colaboran de forma grupal en el desarrollo del diseño del juego

Ideo Digital

kodea BHP Foundation

APRENDO A PROGRAMAR

PLAN DIFERENCIADO DE MATEMÁTICAS III Y IV MEDIO