Universidad
Rey Juan Carlos

Gíua docente: uso de Dr. Scratch para el desarrollo del pensamiento computacional

Gregorio Robles, Jesús Moreno, María Luz Aguado, Eva Hu

Índice

LICENCIA

FIXME: texto de la licencia con la que compartimos el documento.

1 INTRODUCCIÓN

En los últimos años estamos presenciando un movimiento global que defiende que el pensamiento computacional debería ser incluido en la formación de todos los niños y niñas, ya que no solo representa un ingrediente vital del aprendizaje de la ciencia, la tecnología, la ingeniería y las matemáticas, sino que es una competencia fundamental para una vida plena en la sociedad digital hacia la que nos dirigimos.

PERO, ¿QUÉ ES EL PENSAMIENTO COMPUTACIONAL?

Jeannette Wing, en su artículo *Computational Thinking*[1] establece que el "pensamiento computacional implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, haciendo uso de los conceptos fundamentales de la informática". Es decir, que la esencia del pensamiento computacional es pensar como lo haría un informático cuando nos enfrentamos a un problema, de manera que podamos aprovechar la potencia de los ordenadores para resolverlos.

Otras definiciones de pensamiento computacional han ido surgiendo en la literatura científica desde entonces. Entre las más aceptadas se encuentran la de Aho[2] y la de la Royal Society[3]:

- El pensamiento computacional es el proceso que permite formular problemas de forma que sus soluciones pueden ser representadas como secuencias de instrucciones y algoritmos.
- El pensamiento computacional es el proceso de reconocimiento de aspectos de la informática en el mundo que nos rodea, y aplicar herramientas y técnicas de la informática para comprender y razonar sobre los sistemas y procesos tanto naturales como artificiales.

Una iniciativa muy interesante en relación a la definición del pensamiento computacional es la promovida por la Sociedad Internacional de la Tecnología en la Educación (ISTE) y la Asociación de Profesores de Informática (CSTA), que han colaborado con líderes del mundo de la investigación y la educación superior, la industria y la educación primaria y secundaria para desarrollar una definición operativa[4] que describa con precisión sus características esenciales

[1] https://www.cs.cmu.edu/ CompThink/papers/Wing06.pdf
[2] http://comjnl.oxfordjournals.org/content/55/7/832.abstract
[3] https://royalsociety.org/education/policy/computing-in-schools/report/
[4] http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

y ofrezca un marco de trabajo y un vocabulario común con el que los profesionales de la educación puedan trabajar.

Según esta definición operativa, el pensamiento computacional es un proceso de resolución de problemas que incluye las siguientes características:

- Formular problemas de forma que se permita el uso de un ordenador y otras herramientas para ayudar a resolverlos.
- Organizar y analizar lógicamente la información.
- Representar la información a través de abstracciones como los modelos y las simulaciones.
- Automatizar soluciones haciendo uso del pensamiento algorítmico (estableciendo una serie de pasos ordenados para llegar a la solución).
- Identificar, analizar e implementar posibles soluciones con el objetivo de lograr la combinación más efectiva y eficiente de pasos y recursos.
- Generalizar y transferir este proceso de resolución de problemas para ser capaz de resolver una gran variedad de familias de problemas.

PROGRAMACIÓN INFORMÁTICA Y PENSAMIENTO COMPUTACIONAL

Aunque el pensamiento computacional puede trabajarse desde muchas disciplinas e incluso sin necesidad de contar con dispositivos electrónicos, diversas investigaciones demuestran que la programación informática es una muy buena herramienta para desarrollar esta competencia. (FIXME referencia)

Por consiguiente, responsables educativos de todo el mundo han comenzado a incluir la programación en los curriculum nacionales y regionales. Los ejemplos con mayor repercusión han sido, en el panorama internacional, los de Inglaterra, con una nueva asignatura “Computing” obligatoria desde primero de primaria, y Estonia, donde la programación se usa de manera transversal para trabajar diversas asignaturas de primaria y secundaria; y en el caso de España, los de Navarra, que ha incluido la programación en 4º y 5º de primaria asociada al área de matemáticas, y la Comunidad de Madrid, que ha incluido contenidos de programación en la asignatura de tecnología de secundaria y ha creado una asignatura optativa de programación en primaria.

Desde la propia Comisión Europea se está urgiendo a los ministros de la Unión a incluir la programación informática en los planes de estudio para que todos los niños y niñas tengan la oportunidad de desarrollar su pensamiento computacional desde la escuela, ya que estan convencidos de la importancia de esta competencia para la competitividad y la innovación de nuestro continente. (FIXME referencia)En consecuencia, en los próximos años veremos cómo

la programación se incluye de forma paulatina en los curriculum de todos los páises europeos, muy probablemente desde la educación primaria.

CÓMO TRABAJAR Y EVALUAR EL PENSAMIENTO COMPUTACIONAL

Sin duda alguna, la inclusión de actividades de programación con `Scratch`, un lenguaje de programación visual desarrollado específicamente para niños a partir de 6 años, se está implantando en todo el mundo como el estándar para introducir la programación y trabajar el pensamiento computacional en la educación. De hecho, en el momento de escribir este texto, hay más de 8 millones de usuarios registrados en la web de `Scratch` y más de 11 millones de proyectos compartidos.

No obstante, aunque es posible encontrar rúbricas preparadas por distintas entidades educativas para evaluar el pensamiento computacional del alumnado a partir de los proyectos `Scratch` desarrollados por los aprendices, no existen apenas herramientas que permitan automatizar parte de este proceso para ayudar a los docentes. En consecuencia, muchos docentes tienen problemas para poder estudiar en profundidad los proyectos de sus alumnos y poder sacar conclusiones para, por ejemplo, aconsejar a sus alumnos acerca de otros bloques que podrían incorporar a sus programas, formar grupos de estudiantes para trabajar un concepto específico que no parecen haber comprendido completamente, o plantearles proyectos avanzados para desarrollar un aspecto concreto una vez alcanzado un determinado nivel. Y este es el motivo que nos llevó a crear la herramienta `Dr. Scratch`, con el objetivo fundamental de asistir a los docentes en el proceso de enseñanza y evaluación de esta competencia.

DR. SCRATCH Y EL PENSAMIENTO COMPUTACIONAL

La herramienta `Dr. Scratch` permite, entre otras funcionalidades, evaluar el grado de desarrollo del pensamiento computacional a partir de un programa `Scratch`. Las dimensiones evaluadas son las siguientes:

- Abstracción y descomposición de problemas
- Nociones de algoritmia y control del flujo de los programas
- Pensamiento lógico
- Paralelismo
- Sincronización
- Representación de la información
- Interactividad con el usuario

2 USANDO DR. SCRATCH

FIXME: quiz?s aqu? podr?amos meter un par de capturas explicando c?mo se analizan proyectos con Dr. Scratch.

3 DESARROLLANDO LOS DISTINTOS ASPECTOS DEL PENSAMIENTO COMPUTACIONAL CON DR. SCRATCH

Para cada una de las dimensiones del pensamiento computacional que `Dr. Scratch` evalúa, la herramienta asigna una puntuación entre 0 y 3 puntos, en función del grado de desarrollo demostrado en la programación del proyecto analizado. Para aquellas dimensiones en las que existe margen de mejora, `Dr. Scratch` ofrece información para conocer nuevas posibilidades y seguir mejorando cada uno de estos aspectos.

3.1 ABSTRACCIÓN Y DESCOMPOSICIÓN DE PROBLEMAS

La capacidad de abstracción y descomposición de problemas te ayuda a dividir un problema en partes más pequeñas que serán más fáciles de comprender, programar y depurar.

3.1.1 ¿CÓMO CONSEGUIR 1 PUNTO?

Cuando se comienza a programar con `Scratch`, en ocasiones puede parecer que la solución más sencilla es programar todo el comportamiento de un personaje en un único programa. Sin embargo, lo ideal es que el comportamiento del personaje sea controlado por diferentes programas y que cada uno de estos programas se ocupe de una cuestión concreta. Veamos un ejemplo:

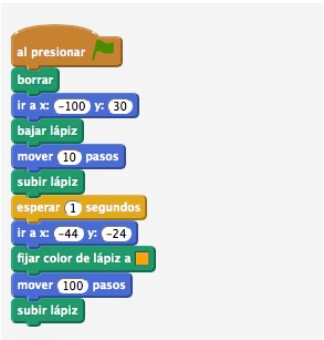

FIGURA 1: FRAGMENTO DE CÓDIGO CON NIVEL 0 EN ABSTRACCIÓN.

Este proyecto, que pinta un dibujo en la pantalla, ha sido programado en un único programa que dibuja las dos líneas que componen el dibujo. Aunque es una opción válida, una opción más sencilla de programar y de mantener es dividir el programa en dos partes, de forma que tengamos dos programas diferentes, uno para pintar la primera línea y otro para pintar la segunda:

FIGURA 2: FRAGMENTO DE CÓDIGO CON NIVEL 1 EN ABSTRACCIÓN.

De este modo, si queremos por ejemplo realizar alguna modificación en una de las líneas dibujadas, es mucho más sencillo saber a qué parte del programa tenemos que dirigirnos para llevar a cabo los cambios.

3.1.2 ¿CÓMO CONSEGUIR 2 PUNTOS?

`Scratch` permite crear nuevos bloques definidos por los usuarios que se componen de una secuencia de instrucciones. Estas abstracciones permiten crear programas más sencillos de leer, de programar y de mantener. Veamos un ejemplo:

FIGURA 3: FRAGMENTO DE CÓDIGO CON BLOQUES REPETIDOS.

Este proyecto `Scratch` dibuja dos líneas naranjas de diferente longitud en la pantalla. En lugar de repetir el código 2 veces, tal como vemos en el ejemplo, es posible definir un bloque ?PintaNaranja? que se compone de los bloques que pintan una línea naranja en la pantalla y al

que se le puede indicar cuál es la longitud de la línea. Para ello hay que irse a la categoría ?Más Bloques? y pulsar en el botón Crear un bloque:

FIGURA 4: FUNCIÓN PROPIA PARA EVITAR REPETIR CÓDIGO.

Una vez definido el bloque ?PintaNaranja? es posible utilizarlo en cualquier programa del proyecto, tal como vemos a continuación:

FIGURA 5: USANDO NUESTRA PROPIA FUNCIÓN.

De este modo, evitamos la repetición de código, lo que hace que nuestros proyectos sean más fáciles de programar y mantener. Tal como se puede observar, la primera vez que se usa el bloque PintaNaranja se le indica una longitud de 100 pasos, mientras que la segunda vez la longitud es de 200 pasos.

3.1.3 ¿CÓMO CONSEGUIR 3 PUNTOS?

En algunos proyectos `Scratch` queremos tener muchos personajes iguales que realizan exactamente las mismas acciones. La primera idea que se nos ocurre para conseguirlo es crear un personaje, programar todo su comportamiento y, una vez que está listo, crear tantas copias como necesitemos. Por tanto, si queremos 20 marcianitos, hay que crear 20 objetos iguales. Sin embargo, ¿qué ocurriría si quiero realizar un cambio en un programa? Tendría que ir objeto por objeto realizando esa modificación.

Para este tipo de situaciones es preferible utilizar clones, un tipo de abstracción que nos ayuda a poder programar un solo objeto, y crear de forma dinámica copias exactas con el mismo comportamiento. Veamos cómo funcionan con un ejemplo. Imagina que queremos simular que está nevando en un proyecto. Podemos dibujar un objeto que sea un copo de nieve, y una vez que comience la ejecución del proyecto, ir creando clones constantemente que aparezcan en la parte superior de la pantalla y vayan cayendo hasta la parte inferior:

FIGURA 6: USANDO CLONES.

De este modo, tan solo programando un personaje podemos tener infinitos clones, que se crean en un momento determinado de la ejecución del proyecto y se borran cuando ya no son necesarios.

En resumen... Aprender a abstraer ayuda a ver grandes problemas difíciles de resolver en pequeños problemas de fácil solución, favoreciendo, además, a que se desarrolle un proyecto de una forma más eficiente.

EJERCICIO

Vamos a proponer un ejercicio para practicar la abstracción. Construye un programa en Scratch en el que el personaje tenga que pintar las letras A, B y C.

3.2 NOCIONES DE ALGORITMIA Y CONTROL DEL FLUJO DE LOS PROGRAMAS

Las instrucciones relacionadas con las nociones algorítmicas de control de flujo pueden ayudarte a controlar el comportamiento de tus personajes, haciendo, por ejemplo, que repitan ciertos bloques un número de veces concreto o que lo repitan hasta que se produzca una situación.

3.2.1 ¿CÓMO CONSEGUIR 1 PUNTO?

La forma más básica de controlar el comportamiento de tus personajes es creando un programa compuesto por un conjunto de bloques que se ejecutan uno detrás de otro, tal como vemos en la imagen:

FIGURA 7: FRAGMENTO DE CÓDIGO CON UN SOLO FLUJO.

¿Cómo funciona este programa? Cuando el usuario presione sobre la bandera verde se ejecutarán uno detras de otro todos los bloques que se han incluido en el programa. Comenzará por el primero 'decir soy un gato que baila por 2 segundos', luego ejecutará 'mover 10 pasos', a continuación 'girar 15 grados a la derecha', y así hasta llegar al último bloque del programa.

3.2.2 ¿CÓMO CONSEGUIR 2 PUNTOS?

En ocasiones, cuando queremos que un conjunto de bloques se repita constantemente, en lugar de repetir los mismos bloques una y otra vez, es posible utilizar instrucciones de repetición que permiten conseguir el mismo efecto, pero de forma más cómoda y manejable. Veamos un par de ejemplos:

FIGURA 8: FRAGMENTOS DE CÓDIGO CON DOS FLUJOS DE EJECUCIÓN.

¿Cómo funciona estos bloques de control? En el bloque 'repetir', cuando la ejecución llega a este punto se repetirán tantas veces como se haya indicado en el parámetro del bloque 'repetir', en el ejemplo serían 3 veces, todos los bloques incluidos dentro de este bloque. Podemos ver que el bloque 'repetir' tiene una flechita en la parte inferior que indica que cuando se terminan de ejecutar los bloques se vuelve otra vez para arriba. El bloque 'por siempre' funciona igual que el 'repetir' pero sin terminar nunca de ejecutar los bloques que contiene en su interior.

3.2.3 ¿CÓMO CONSEGUIR 3 PUNTOS?

En algunas ocasiones no sabemos previamente el número de veces que queremos que un conjunto de bloques se ejecute, ya que esto depende de una determinada situación. En estas ocasiones, el bloque 'repetir hasta que...' es realmente útil. Veamos un ejemplo:

FIGURA 9: USANDO EL BLOQUE 'SI'.

En el ejemplo, el personaje repetirá constantemente la instrucción decir 'No me pillas...' mientras no esté tocando al enemigo. En el momento que se cumpla la condición de la instrucción 'repetir hasta que...', terminará de ejecutarse el bloque contenido en su interior y saltará a la siguiente instrucción, en este caso decir' Me pillaste!'.

En resumen... Aprender sobre algoritmia y control de flujo nos ayuda a esquematizar y a poner orden en las acciones que se realizan en nuestro programa.

EJERCICIO

Vamos a proponer un ejercicio para practicar el control de flujo. Construye un programa en Scratch en el que se guíe paso a paso a un cocinero a la hora de realizar la receta de tu plato favorito.

3.3 PENSAMIENTO LÓGICO

Las instrucciones relacionadas con el pensamiento lógico pueden ayudarte a que tus proyectos sean dinámicos, de forma que se comporten de modo distinto en función de la situación. En las historias, por ejemplo, este tipo de instrucciones no son tan importantes, ya que suelen tener una estrucutura lineal que siempre queremos que se ejecute del mismo modo, pero en otro tipo de proyectos, como los videojuegos, son fundamentales para ejecutar acciones diferentes dependiendo de la situación.

3.3.1 ¿CÓMO CONSEGUIR 1 PUNTO?

El bloque más básico con el que puedes comenzar a trabajar el pensamiento lógico es éste:

FIGURA 10: USANDO EL BLOQUE 'SI'.

¿Cómo funciona este bloque? Cuando la ejecución llega a este punto, se evalúa la condición que aparece en el bloque, y si ésta es verdadera, se ejecuta el conjunto de bloques que están dentro del si. En el ejemplo, si el personaje está tocando el color azul diría ?Llegué a la meta?.

3.3.2 ¿CÓMO CONSEGUIR 2 PUNTOS?

Ahora que conoces las instrucciones si, quizás puedas comenzar a utilizar bloques si/sino, que pueden ser muy prácticos en muchos tipos de proyectos. Los bloques si/sino son muy parecidos a los bloque si, pero contienen dos conjuntos de bloques en su interior.

FIGURA 11: USANDO EL BLOQUE 'SI, SI NO'.

¿Cómo funciona este bloque? Cuando la ejecución llega a este punto, se evalúa la condición que aparece en el bloque, y si ésta es verdadera, se ejecuta el conjunto de bloques que están dentro del si. En el ejemplo, si el personaje está tocando el color azul diría ?Llegué a la meta?,pero si no está tocando el color morado diría ?Aún no he llegado a la meta?.

3.3.3 ¿CÓMO CONSEGUIR 3 PUNTOS?

Para tomar decisiones en algunos proyectos en ocasiones se requiere evaluar más de una condición al mismo tiempo para saber qué hay que ejecutar. En esas situaciones es muy útil utilizar operaciones lógicas que permiten combinar las condiciones. Las operaciones lógicas disponibles en Scratch son Y, O y No.

FIGURA 12: EVALUANDO CONDICIONES.

La operación Y es verdadera cuando las dos condiciones de evaluadas son verdaderas. La operación O es verdadera cuando una de las dos condiciones evaluadas son verdaderas. Y la

operación No valdrá lo contrario de la condición, es decir, si la condición es verdadera, No devuelve falso y viceversa.

En resumen... El pensamiento lógico nos ayuda a ejecutar cierto código si se cumplen unas determinadas condiciones, evitando ejecutarlo cuando no es necesario.

EJERCICIO

(PENSAR EJERCICIO)

3.4 PARALELISMO

En la mayoría de las creaciones Scratch se requiere un cierto nivel de paralelismo. Pero, ¿qué es el paralelismo? El paralelismo es la posibilidad de que varias cosas ocurran al mismo tiempo. Por ejemplo, que dos personajes realicen una acción al mismo tiempo, o que un personaje haga varias cosas a la vez.

3.4.1 ¿CÓMO CONSEGUIR 1 PUNTO?

La forma más básica y más evidente de conseguir paralelismo es tener varios programas que comienzan con ?Al presionar la bandera verde?:

FIGURA 13: BLOQUE .ªL PULSAR BANDERA VERDE".

De este modo, cuando el usuario pinchase sobre la bandera verde, todos los programas que comienzan con este bloque comenzarían a ejecutarse al mismo tiempo, o en paralelo, como también puede decirse. Podrías tener varios programas que comiencen con este bloque en un mismo personaje, si quieres que haga varias cosas a la vez, o en programas de diferentes personajes, si quieres que todos comiencen a realizar una acción al comenzar la ejecución.

3.4.2 ¿CÓMO CONSEGUIR 2 PUNTOS?

Otra forma de conseguir paralelismo en tus programas es haciendo que ocurran varias cosas cuando el usuario presiona una tecla o hace click sobre un objeto. Veamos un par de ejemplos:

FIGURA 14: CÓDIGO QUE SE EJECUTA EN PARALELO.

¿Cómo funcionan estos bloques de control? En las dos primeras figuras de la izquierda vemos fragmentos de código iguales que pertenecen a dos personajes distintos, de manera que, cuando se pulsa una tecla, realizan una determinada acción. Por tanto, cuando el usuario pulse la tecla a, en este caso, tanto un personaje como el otro ejecutarán al mismo tiempo ?decir A por 2 segundos?. En el último ejemplo vemos que un personaje tiene dos programas que comienzan con ?al clickear este objeto?. Por tanto, cuando el usuario haga click sobre este personaje, ambos programas comenzarán a ejecutarse al mismo tiempo, en paralelo.

3.4.3 ¿CÓMO CONSEGUIR 3 PUNTOS?

Existen varios eventos más que permiten conseguir paralelismo:

FIGURA 15: DISPARADORES DE EVENTOS.

Como ves, podrías crear varios programas que comenzaran a ejecutarse al cambiar el fondo a un determinado escenario, o al recibir un mensaje concreto, o cuando el volumen del ambiente sea superior a un determinado umbral, cuando el movimiento del vídeo se mayor que un número de pixels concreto o cuando el cronómetro haya superado el valor que tú quieras. Por tanto, existen muchas posibilidades para conseguir que en tus programas ocurran cosas al mismo tiempo. ¿Te animas a probar algunas de ellas?

En resumen... El paralelismo nos permite ejecutar varias acciones a la vez en un mismo personaje y/o en varios capturando determinados eventos(teclado, ratón, sonido, vídeo, mensajes...).

EJERCICIO

(PENSAR EJERCICIO)

3.5 SINCRONIZACIÓN

Las instrucciones relacionadas con la sincronización permiten organizar a nuestros personajes para que las cosas ocurran en el orden que nosotros deseamos.

3.5.1 ¿CÓMO CONSEGUIR 1 PUNTO?

La forma más sencilla de sincronizar el comportamiento de tus personajes es utilizando un bloque 'esperar', que hace que el personaje espere el número de segundos que escribamos como parámetro del bloque. Veamos un ejemplo de cómo puede utilizarse este bloque para sincronizar a dos personajes:

FIGURA 16: USANDO BLOQUE ES 'ESPERAR'.

En este caso se han utilizado los bloques esperar' para sincronizar a estos dos personajes para que mantengan una conversación, de forma que mientras uno habla, el otro espera, y viceversa. Fíjate en que el número de segundos que cada personaje espera, son iguales al número de segundos que el otro personaje habla, de manera que nunca hablan al mismo tiempo.

3.5.2 ¿CÓMO CONSEGUIR 2 PUNTOS?

La sincronización mediante bloques 'esperar' es muy sencilla cuando los programas son pequeños y tenemos pocos personajes, pero cuando son más grandes, o cuando tenemos varios personajes, o cuando las condiciones para generar una reacción no pueden ser medidas previamente, es más eficiente utilizar otros modos de sincronización como los mensajes. Veamos un ejemplo:

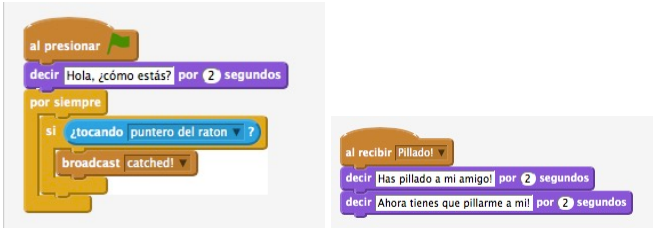

FIGURA 17: USANDO ENVÍO DE MENSAJES.

¿Cómo funcionan estos bloques de sincronización? Cuando se produce una situación en un personaje que queremos que provoque una reacción en otro personaje, podemos hacer uso del envío de mensajes. En el ejemplo, cuando el ratón toca al gato, se envía el mensaje 'Pillado', que será enviado a todos los personajes del proyecto. Así, cuando la mariposa recibe el mensaje 'Pillado', se ejecutan las instrucciones incluidas bajo el bloque 'al recibir Pillado'. Por tanto, cuando el usuario toque con el ratón al gato, la mariposa dirá: 'Has pillado a mi compañero. Ahora tienes que pillarme a mí'.

3.5.3 ¿CÓMO CONSEGUIR 3 PUNTOS?

Además del envío de mensajes, es posible sincronizar personajes para que las cosas ocurran en el orden que nosotros deseamos utilizando otro tipo de bloques, como 'esperar hasta que' o 'cuando el fondo cambie a...'.

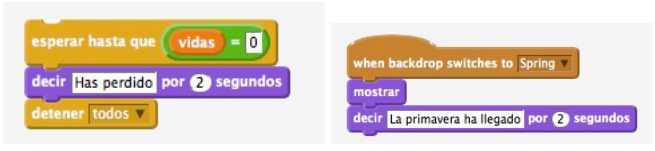

FIGURA 18: OTROS BLOQUES DE SINCRONIZACIÓN.

En el primer ejemplo, cuando la ejecución del programa llegue a ese punto se detendrá hasta que se cumpla la condición, en este caso hasta que vidas sea igual a 0. Cuando vidas valga 0,

entonces continuará la ejecución del resto de bloques, en este caso 'decir Game Over' y 'detener todos', lo que finalizará la ejecución del proyecto. En el segundo ejemplo, en el momento en el que el escenario cambie al fondo con el nombre 'Primavera', el personaje ejecutará los bloques colocados a continuación de la instrucción 'cuando el fondo cambia a 'Primavera', es decir, que se mostrará y dirá '°Ha llegado la primavera'.

En resumen... La sincronizaci?n permite que las cosas ocurran en el orden que nosotros queramos.

EJERCICIO

Realizar un programa en Scratch en el que se envíe mensajes entre dos personajes.

3.6 REPRESENTACIÓN DE LA INFORMACIÓN

Cada proyecto Scratch necesita un conjunto de información sobre los personajes para poder ejecutarse correctamente. Por ejemplo, necesita conocer la posición de cada personaje, la dirección a la que está apuntando, su tamaño, etc. Además, los programadores podemos crear nuevos depósitos de información para guardar otro tipo de datos, como el nivel en el que nos encontramos, el tiempo transcurrido, la puntación, las vidas, las recompensas recogidas...

3.6.1 ¿CÓMO CONSEGUIR 1 PUNTO?

Cada personaje de un proyecto Scratch tiene una serie de características, o atributos, que en cada momento de ejecución del proyecto tienen un valor, y que pueden ser modificadas por los programas. Por ejemplo, un personaje tiene un tamaño determinado que puede ser modificado por el bloque 'cambiar tamaño por 10', haciendo que el personaje aparezca un poco más grande. A continuación mostramos la lista de atributos que tiene cada personaje y los bloques que pueden utilizarse para modificarlos:

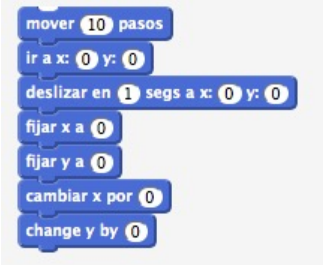

FIGURA 19: POSICIÓN.

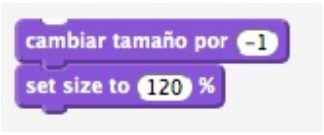

FIGURA 20: TAMAÑO.

FIGURA 21: ORIENTACIÓN.

FIGURA 22: DISFRAZ.

FIGURA 23: VISIBILIDAD.

3.6.2 ¿CÓMO CONSEGUIR 2 PUNTOS?

Además de modificar los atributos de los personajes, los programadores podemos utilizar otros mecanismos para almacenar información en un proyecto Scratch. Uno de estos mecanismos son las variables, que permiten almacenar un valor para guardar distintos tipos de datos que podemos necesitar: en qué nivel nos encontramos, cuántas vidas nos quedan, cuántos puntos llevamos, cómo se llama el usuario... Para crear una variable hay que irse a la categoría de Datos, pinchar sobre 'Crear una variable' y darle un nombre. En el ejemplo que mostramos a continuación le hemos dado el nombre 'Puntos', ya que la utilizaremos para guardar el número de puntos que consigue nuestro personaje.

FIGURA 24: VARIABLES.

En el prorgrama de la izquierda asignamos el número de puntos con los que queremos que el personaje comience la partida, en este caso 0, para lo que utilizamos el bloque 'fijar Puntos a 0'. Sin embargo en el programa de la derecha lo que hacemos es que cuando el personaje toque el color azul, que podría ser el de un objetivo, le vamos a sumar un punto, para lo que utilizamos el bloque 'cambiar Puntos por 1'. En este caso, el bloque 'cambiar' comprobaría el valor actual de la variable Puntos y le sumaría 1: si vale 0, le sumo 1 y ahora vale 1; si vale 1, le sumo 1 y ahora vale 2...

3.6.3 ¿CÓMO CONSEGUIR 3 PUNTOS?

Además de las variables, Scratch permite utilizar otro tipo de datos para guardar información de un proyecto: las listas. Las listas permiten almacenar más de un valor al mismo tiempo, por lo que son ideales para guardar las recompensas que han sido recuperadas por un personaje, o un conjunto de nombres, por plantear un par de ejemplos. Para crear una lista, hay que irse a la categoría de Datos, pinchar sobre 'Crear una lista' y darle un nombre. En el ejemplo, nosotros hemos llamado a nuestra lista 'Estudiantes':

FIGURA 25: OTROS BLOQUES DE SINCRONISMO.

¿Qué hemos hecho con este programa? Lo primero que hacemos al comenzar la ejecución es borrar todos los elementos de la lista Estudiantes, para vaciarla completamente y que no aparezcan elementos de una ejecución anterior del proyecto. A continuación le pedimos tres veces al usuario que introduzca un nombre, y cada nombre introducido lo insertamos en la última posición de la lista Estudiantes. ¿Qué podemos hacer con la lista? Podríamos, por ejemplo, utilizarla para sacar a un voluntario a realizar un ejercicio a la pizarra digital. Fijaos en las nuevas instrucciones que añadimos al programa:

FIGURA 26: OTROS BLOQUES DE SINCRONISMO.

Hemos generado un número al azar entre 1 y 3, que son los elementos que hay en la lista, y hemos seleccionado al elemento que ocupa esa posición en la lista. Unas veces cogeremos el nombre que está en la posición 1, otras al que está en la posición 2 y otras al de la posición 3. Las listas tienen muchas posibilidades y muchas operaciones disponibles... ¿te animas a investigar cómo funcionan?

En resumen... El sincronismo permite que las cosas ocurran en el orden que nosotros queramos.

EJERCICIO

Realizar un programa en Scratch en el que se envíe mensajes entre dos personajes.

3.7 INTERACTIVIDAD CON EL USUARIO

4 PROMOVIENDO BUENOS HÁBITOS DE PROGRAMACIÓN CON DR. SCRATCH

4.1 NOMBRADO DE OBJETOS

4.2 CÓDIGO MUERTO

4.3 INICIALIZACIÓN DE LOS ATRIBUTOS DE LOS PERSONAJES

4.4 REPETICIÓN DE CÓDIGO

5 CONCLUSIONES FINALES

Este documento presenta una gu□a sobre el uso de Dr. Scratch como herramienta de apoyo para desarrollar el pensamiento computacional, que ha sido dise□ada con el objetivo de ayudar a los docentes a comprender las diferentes dimensiones que componen esta competencia, y ofrecer ideas y estrategias para su trabajo en el aula con el alumnado.

Esta gu□a, como todo el proyecto Dr. Scratch, se comparte con una licencia libre y se encuentra disponible en un repositorio p□blico[5], de manera que animamos al lector a compartir sus opiniones y cr□ticas para que podamos seguir mejorando el documento, y del mismo modo, le animamos a participar activamente en el desarrollo de la pr□xima versi□n del mismo.

A SOLUCI□N A LOS EJERCICIOS PROPUESTOS

FIXME: Quiz□s podr□an crearse estudios por cada uno de los ejercicios propuestos (que en principio solo tendr□an un proyecto programado por nosotros con la soluci□n), de forma que animemos a la gente a compartir sus soluciones a□adiendo sus proyectos al estudio.

[5] https://github.com/kgblll/Documentation/tree/master/LibroDocentes