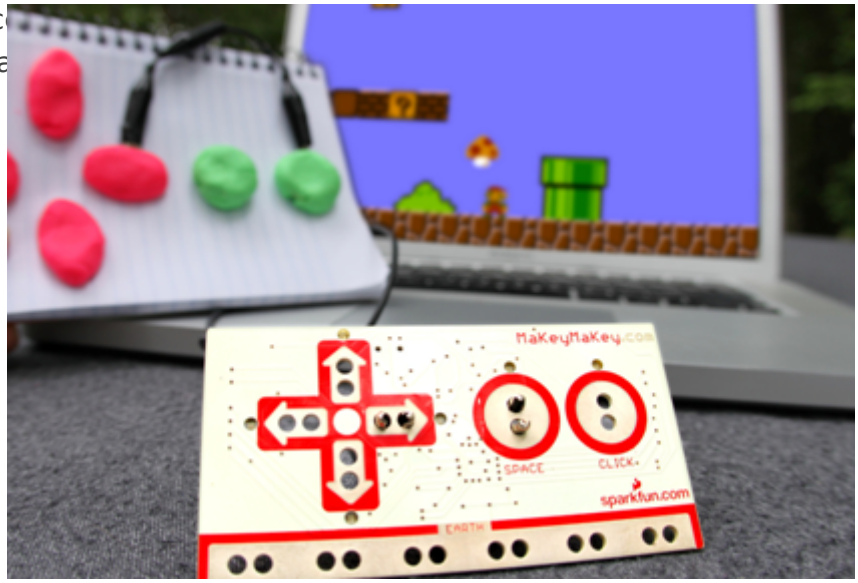
Scratch Avanzado Y Makey Makey

Introducción

Este curso es **UNA CONTINUACIÓN** **DEL CURSO BÁSICO DE SCRATCH** por lo que precisa de sus conceptos básicos.

Objetivos

- Seguir con la gamificación en Scratch: Aprender jugando con proyectos e introducir instruccionés más complejas.
- Conocer Makey Makey y sus posibilidades.
- Conc
- Herra



CENTRO ARAGONÉS de TECNOLOGÍAS para la EDUCACIÓN

CATEDU

{% include "git+https://github.com/catedu/robotica.git/README.md" %}

Pensamiento computacional

Oferta de formación en Pensamiento computacional del Centro Aragonés de Tecnologías para la Educación.

https://view.genial.ly/5c546dc28805472c3451861a

> Tenemos un **grupo Telegram Robótica Educativa en Aragón**, si estás interesado en unirte, envía un mensaje por Telegram (obligatorio) a CATEDU 623197587
>
> https://t.me/catedu_es y te añadimos en el grupo

Robótica y accesibilidad

1.- Introducción

Durante mucho tiempo la robótica fue patrimonio de personas y/o instituciones con alta capacidad económica (podían adquirir las placas con microcontroladores comerciales) y capacidad intelectual (podían entender y programar el funcionamiento de las mismas) siempre dentro de los límites establecidos por las marcas comerciales y lo que pudieran “desvelar” de su funcionamiento, vigilando siempre que la competencia no “robara” sus secretos y “copiara” sus soluciones.

Todo esto saltó por los aires en torno a 2005 con la irrupción de un grupo de profesores y estudiantes jóvenes, que decidieron romper con esta dinámica, tratando de poner a disposición de su alumnado microcontroladores económicamente accesibles y que les permitieran conocer su funcionamiento, sus componentes, e incluso replicarlos y mejorarlos. Nacía **Arduino** y el concepto de **Hardware Open Source**. Detrás de este concepto se encuentra la **accesibilidad universal.** En un proyecto Open Source todo el mundo puede venir, ayudar y contribuir, minimizando barreras económicas e intelectuales.

Arduino traslada al hardware un concepto ya muy conocido en el ámbito del software, como es el **software open source o software libre.**

Software libre

Cuando los desarrolladores de software terminan su creación, tienen múltiples posibilidades de ponerlo a disposición de las personas, y lo hacen con condiciones específicas especificadas en una licencia. Esta licencia es un contrato entre el creador o propietario de un software y la persona que finalmente acabará utilizando este software. Como usuarios, es nuestro deber conocer las condiciones y permisos con las que el autor ha licenciado su producto, para conocer bajo qué condiciones podemos instalar y utilizar cada programa.

Existen muchas posibilidades de licencias: software privativo, comercial, freeware, shareware, etc.. Nos centraremos aquí en la de software libre.

GNU (https://www.gnu.org) es una organización sin ánimo de lucro que puso una primera definición disponible de lo que es software libre: Software libre significa que los usuarios del software tienen libertad (la cuestión no es el precio). Desarrollaron el sistema operativo GNU para que los usuarios pudiesen tener libertad en sus tareas informáticas. Para GNU, el software libre implica que los usuarios tienen las cuatro libertades esenciales:

1. ejecutar el programa.
2. estudiar y modificar el código fuente del programa.
3. redistribuir copias exactas.
4. distribuir versiones modificadas.

En otras palabras, el software libre es un tipo de software que se distribuye bajo una licencia que **permite a los usuarios utilizarlo, modificarlo y distribuirlo libremente**. Esto significa que los usuarios tienen libertad de ejecutar el software para cualquier propósito, de estudiar cómo funciona el software y de adaptarlo a sus necesidades, de distribuir copias del software a otros usuarios y de mejorar el software y liberar las mejoras al público.

El software libre se basa en el principio de la libertad de uso, y no en el principio de la propiedad. Esto significa que los usuarios tienen la libertad de utilizar el software de la manera que deseen, siempre y cuando no violen las condiciones de la licencia. El software libre es diferente del software propietario, que es el software que se distribuye con restricciones en su uso y modificación. El software propietario suele estar protegido por derechos de autor y solo se puede utilizar bajo los términos y condiciones especificados por el propietario del software.

Recomendamos la visualización de este video para entender mejor el concepto.

https://www.youtube.com/embed/nlDVZ816zoI

Más adelante, entorno a 2015, en Reino Unido, surgiría también la placa **BBC Micro:bit**, con la misma filosofía de popularizar y hacer accesible en este caso al alumnado de ese país la programación y la robótica. También hablaremos de ella.

2.- ARDUINO o LA ROBÓTICA ACCESIBLE

Arduino es una **plataforma de hardware y software libre**.

Hardware libre

Esto significa que tanto la placa Arduino como el entorno de desarrollo integrado (IDE) son de código abierto. Arduino permite a los usuarios utilizar, modificar y distribuir tanto el software como el hardware de manera libre y gratuita, siempre y cuando se respeten las condiciones de

las licencias correspondientes.

El hardware libre es un tipo de hardware cuya **documentación y diseño están disponibles de manera gratuita y libre** para su modificación y distribución. Esto permite a los usuarios entender cómo funciona el hardware y adaptarlo a sus necesidades, así como también crear sus propias versiones modificadas del hardware.

Arduino surge como solución al **elevado precio de los microcontroladores** allá por el año 2005. En el ámbito de la educación, los microcontroladores solo se utilizaban en la etapa universitaria, y su coste era tan elevado que muchos proyectos de fin de carrera se quedaban únicamente en prototipos virtuales ya que las universidades no podían proveer a cada estudiante con un microprocesador, contando además que en el propio proceso de experimentación lo más habitual era que una mala conexión hiciera que se rompieran. Otro **gran inconveniente era la dificultad de la programación**. Cada fabricante entregaba su manual de programación, lo que hacía que de unos a otros no hubiera un lenguaje estándar, y la consecuente dificultad de interpretación. Además, su programación era a bajo nivel en lenguaje máquina. Generar una simple PWM requería una ardua y minuciosa secuenciación que podía llevar varias horas hasta conseguir el resultado deseado. Por este motivo, el enfoque de Arduino desde el principio fue ser Open Source tanto en hardware como en software. El desarrollo del hardware fue la parte más sencilla. Orientado a educación, sufre algunas modificaciones frente a los microprocesadores existentes para hacer más fácil su manejo y accesibilidad a cualquier sensor o actuador. El mayor esfuerzo se entregó en todas las líneas de código que hacían posible que ya no hubiera que programar a bajo nivel gracias al IDE de Arduino que incluía bibliotecas y librerías que estandarizaban los procesos y hacían tremendamente sencillo su manejo. Ahora el alumnado para mover un motor, ya no tenía que modificar las tramas de bits del procesador una a una, sino que bastaba con decir que quería moverlo en tal dirección, a tal velocidad, o a equis grados.

Acabábamos de pasar de unos costes muy elevados y una programación muy compleja a tener una **placa accesible, open source y de bajo coste** que además hacía muy **accesible su programación y entendimiento**, características fundamentales para su implantación en educación, hasta tal punto que su uso ya no era exclusivo de universidades, sino que se extiende a la educación secundaria.

Este hecho es fundamental para el desarrollo del Pensamiento Computacional en el aula observándose que su accesibilidad y beneficios son tales, que alcanzan a **centros con alumnado de toda tipología** como la aplicación del pensamiento computacional y robótica en aulas con alumnos de necesidades especiales. Una vez más, aparece el concepto de accesibilidad asociado a esta filosofía Open Source.

A este respecto, recomendamos la lectura de este interesante blog, que tiene por título: ROBOTIQUEAMOS...” Experiencia de aproximación a la robótica en Educación Especial (CPEE ÁNGEL RIVIÈRE). También recomendamos los trabajos robótica en Educación Especial (CPEE ÁNGEL RIVIÈRE): http://zaragozacpeeangelriviere.blogspot.com/search/label/ROB%C3%93TICA

Igualmente, la aparición de Arduino supone una gran facilidad para la aplicación de la robótica y la programación en la atención temprana, donde son numerosas sus aplicaciones desde ayudar a mitigar el déficit de atención en jóvenes autistas, hasta ayudar a socializar a los alumnos con dificultades para ello, o ayudar a alumnos de altas capacidades a desarrollar sus ideas.

Por otro lado su accesibilidad económica lo ha llevado a popularizarse en países de **todo el mundo**, especialmente en aquellos cuyos sistemas educativos no disponen en muchas ocasiones de recursos suficientes, lo que supone en la práctica una **democratización del conocimiento y superación de brecha digital.**

Filosofía del Arduino ver vídeo

Arduino y su IDE son la primera solución que aparece en educación con todas las ventajas que hemos enumerado, y esto hace que todos los nuevos prototipados y semejantes tengan algo en común, siempre son compatibles con Arduino

Para entender bien la filosofía de Arduino y el hardware libre, os recomendamos este documental de 30 minutos. Arduino the Documentary

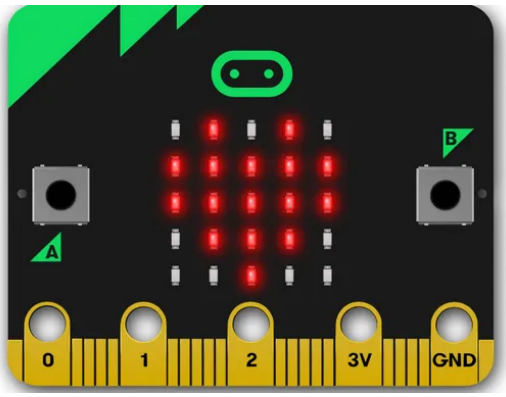
https://player.vimeo.com/video/18390711?h=b5844e7753

Scratch: software libre para el desarrollo del pensamiento computacional

Scratch es un lenguaje de programación visual desarrollado por el grupo Lifelong Kindergarten del MIT Media Lab. Scratch es un software libre. Esto significa que está disponible gratuitamente para todos y que se distribuye bajo una licencia de software libre, la Licencia Pública General de Massachusetts (MIT License). Esta licencia permite a los usuarios utilizar, modificar y distribuir el software de manera libre, siempre y cuando se respeten ciertas condiciones. Entre otras cosas, la licencia de Scratch permite a los usuarios utilizar el software para cualquier propósito, incluyendo fines comerciales. También permite modificar el software y distribuir las modificaciones, siempre y cuando se incluya una copia de la licencia y se indique que el software ha sido modificado. En resumen, Scratch es un software libre que permite a los usuarios utilizar, modificar y distribuir el software de manera libre y gratuita, siempre y cuando se respeten las condiciones de la licencia. De hecho, gracias a que está licenciado de esta forma, han surgido decenas de variaciones de Scratch para todo tipos de propósitos, eso sí, siempre educativos y relacionados con las enseñanzas de programación y robótica

3. BBC micro:bit y la Teoría del Cambio

BBC micro:bit, a veces escrito como Microbit o Micro Bit, es un pequeño ordenador del tamaño de media tarjeta de crédito, creado en 2015 por la BBC con el fin de promover el desarrollo de la robótica y el pensamiento computacional entre la población escolar del Reino Unido. Actualmente s es de escolares de 7 a 16 años de más de 60 países.



Tarjeta BBC micro:bit V1. Fuente: https://microbit.org. CC BY-SA 4.0.

Aunque el proyecto fue iniciado por la BBC, su desarrollo fue llevado a cabo por 29 socios tecnológicos de primera línea. Por ejemplo, la implementación del Bluetooth integrado en la tarjeta corrió a cargo de la fundación propietaria de la marca, Bluetooth SIG, una asociación privada sin ánimo de lucro.

El hardware y el software resultantes son 100% abiertos, y están gestionados por una fundación sin ánimo de lucro que comenzó a funcionar en el año 2016, la Micro:bit Educational Foundation. La fundación basa sus actuaciones en su Teoría del Cambio,

Teoría del cambio y más sobre microbit

Teoría del cambio puede resumirse en tres principios:

- El convencimiento de que la capacidad de comprender, participar y trabajar en el mundo digital es de vital importancia para las oportunidades de vida de una persona joven.
- La necesidad de emocionar y atraer a las personas jóvenes por medio de BBC micro:bit, especialmente a las que podrían pensar que la tecnología no es para ellas.
- Diversificar a los estudiantes que eligen las materias STEM a medida que avanzan en la escuela y en sus carreras, para hacer crecer una fuente diversa de talento, impulsando la equidad social y contribuyendo a crear una tecnología mejor.

Para desarrollar sus principios, la fundación trabaja en tres líneas de acción:

- El desarrollo de hardware y software que contribuyan a despertar el entusiasmo en las personas jóvenes hacia la tecnología y hacia las oportunidades que presenta.
- La creación de recursos educativos gratuitos y fáciles de usar que permitan al profesorado enseñar de forma atractiva y creativa.
- La colaboración con entidades asociadas que compartan una misma visión para ofrecer programas educativos de alto impacto en todo el mundo.

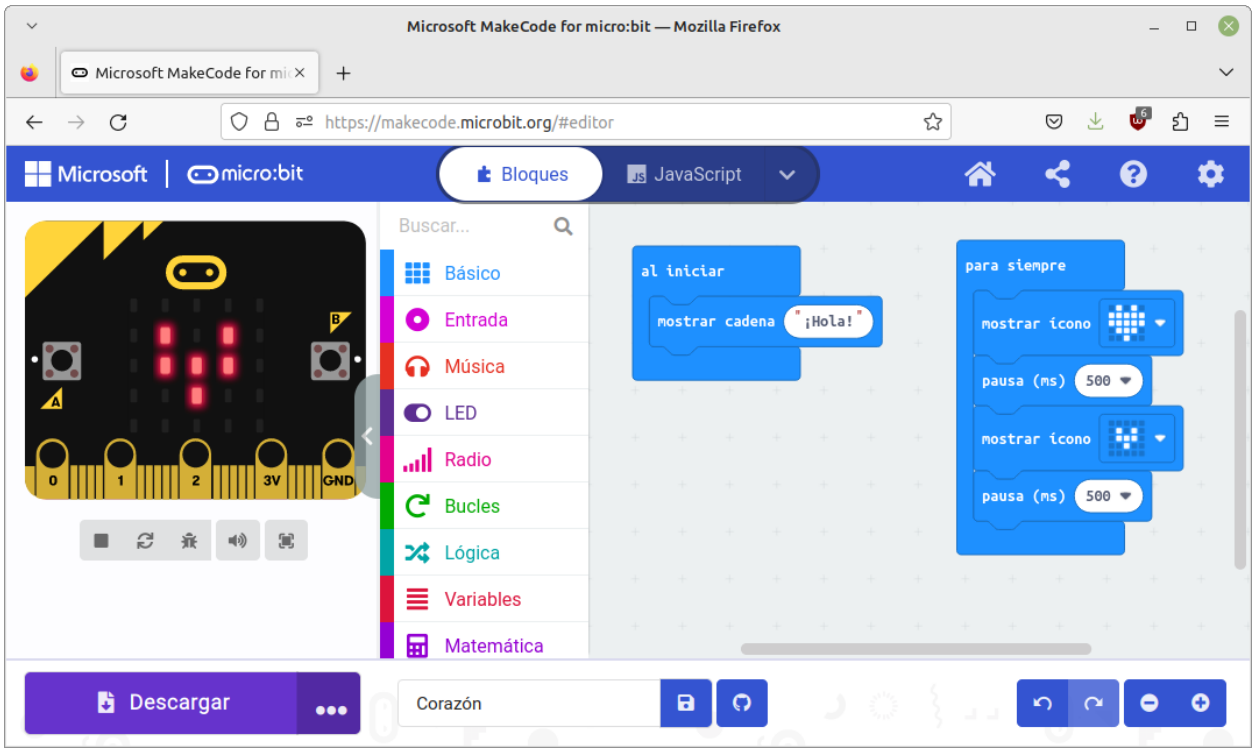
> Uno de los objetivos de la Micro:bit Educational Foundation es llegar a 100 millones de escolares en todo el mundo.

En correspondencia con las líneas de acción y con los principios expuestos, el sistema resultante es muy económico: tanto las placas como los accesorios producidos por terceras empresas tienen un precio muy contenido. Además, dado el carácter abierto del proyecto, están disponibles algunos clones totalmente compatibles, como Elecrow Mbits o bpi:bit. Estos clones son incluso más potentes y económicos que la placa original.

El universo micro:bit destaca por su **alta integración de software y hardware**: basta un clic de ratón para cargar las librerías necesarias para que funcione cualquier complemento robótico, como sensores, pantallas, tarjetas de Internet de las Cosas, robots, casas domóticas, etc.

La programación de la placa se realiza desde un ordenador a través de un navegador cualquiera, estando disponibles **12 lenguajes de programación**. De nuevo, por ser un sistema abierto, existen múltiples soluciones de programación, aunque las más común es

MakeCode.



Captura de pantalla del editor MakeCode, https://makecode.microbit.org/#.

El sitio web MakeCode permite programar con bloques y también en Python y en Java, traduciendo de un lenguaje a otro instantáneamente. No se necesita ningún registro en la plataforma para poder programar.

Los programas también pueden guardarse descargados en el ordenador compilados en código de máquina. Al subir de nuevo el programa al editor, se realiza una decompilación automática al lenguaje de bloques, Python o Java. Los programas guardados en código de máquina se pueden cargar directamente en micro:bit, que en el escritorio de un ordenador se maneja como una simple unidad de memoria USB.

MakeCode contiene además múltiples recursos como tutoriales, vídeos, fichas de programación, cursos para el profesorado, ejemplos y propuestas de proyectos y experimentos, todo ello en varios idiomas y clasificado por edades desde los 7 años.

Otra solución muy usada para programar micro:bit es MicroPython, creada por Python Software Foundation, otra organización sin ánimo de lucro.

MicroCode permite que los más pequeños, a partir de los 6 años de edad, programen micro:bit mediante un sistema de fichas dispuestas en líneas de acción. Están disponibles un tutorial introductorio en 20 idiomas, una guía del usuario y muchos ejemplos. El proyecto es de código abierto.

Micro:bit también es programable en **Scratch** con sólo añadir una extensión al editor.

Todos los entornos de desarrollo descritos disponen de un simulador de micro:bit, por lo que ni siquiera resulta necesario disponer de una tarjeta física para aprender a programar.

Una vez realizada la programación, la placa y sus complementos pueden funcionar desconectados del ordenador por medio de un cargador de móvil, una batería externa o un simple par de pilas alcalinas.

Versiones y características de micro:bit

A pesar de su pequeño tamaño, micro:bit es un sistema potente. Existen dos versiones de la placa. La más moderna, llamada micro:bit V2, tiene las siguientes características:

- Procesador de 64 MHz.
- 512 KB de RAM Flash y 128 KB de RAM.
- Matriz de 5 x 5 LED rojos.
- Dos pulsadores mecánicos y un tercer pulsador de apagado y reset.
- Un pulsador táctil.
- Micrófono y altavoz.
- Acelerómetro y brújula.
- Sensores de luz y de temperatura.
- Comunicación con otras placas por Bluetooth de bajo consumo.
- Alimentación a 3 V o por USB.
- 25 pines de entradas y salidas para conectar motorcitos, sensores, placas de Intenet de las Cosas, robots y, en general, cualquier otro tipo de accesorio.
- 200 mA de intensidad de corriente disponibles en las salidas para alimentar accesorios.

4.- LA IMPORTANCIA DEL OPEN SOURCE / CÓDIGO ABIERTO EN EDUCACIÓN

La creación, distribución, modificación y redistribución del hardware y software libre así como su utilización, están asociados a una serie de valores que deberían ser explicados en la escuela a nuestros alumnos para dar una alternativa a la versión mercantilista de que cualquier creación es creada para obtener beneficios económicos.

En GNU, pusieron especial énfasis en la difusión del software libre en colegios y universidades, promoviendo una serie de valores fundacionales:

Valores GNU

Compartir

El código fuente y los métodos del hardware y software libre son parte del conocimiento humano. Al contrario, el hardware software privativo es conocimiento secreto y restringido. El código abierto no es simplemente un asunto técnico, es un asunto ético, social y político. Es una cuestión de derechos humanos que la personas usuarias deben tener. La libertad y la cooperación son valores esenciales del código abierto. El sistema GNU pone en práctica estos valores y el principio del compartir, pues compartir es bueno y útil para el progreso de la humanidad. Las escuelas deben enseñar el valor de compartir dando ejemplo. El hardware y software libre favorece la educación pues permite compartir conocimientos y herramientas.

Responsabilidad social

La informática, electrónica, robótica... han pasado a ser una parte esencial de la vida diaria. La tecnología digital está transformando la sociedad muy rápidamente y las escuelas ejercen una influencia decisiva en el futuro de la sociedad. Su misión es preparar al alumnado para que participen en una sociedad digital libre, mediante la enseñanza de habilidades que les permitan tomar el control de sus propias vidas con facilidad. El hardware y el software no debería estar bajo el poder de un desarrollador que toma decisiones unilaterales que nadie más puede cambiar.

Independencia

Las escuelas tienen la responsabilidad ética de enseñar la fortaleza, no la dependencia de un único producto o de una poderosa empresa en particular. Además, al elegir hardware y software libre, la misma escuela gana independencia de cualquier interés comercial y evita permanecer cautiva de un único proveedor. Las licencias de hardware y software libre no expiran

Aprendizaje

Con el open source los estudiantes tienen la libertad de examinar cómo funcionan los dispositivos y programas y aprender cómo adaptarlos si fuera necesario. Con el software libre se aprende también la ética del desarrollo de software y la práctica profesional.

Ahorro

Esta es una ventaja obvia que percibirán inmediatamente muchos administradores de instituciones educativas, pero se trata de un beneficio marginal. El punto principal de este aspecto es que, por estar autorizadas a distribuir copias de los programas a bajo costo o gratuitamente, las escuelas pueden realmente ayudar a las familias que se encuentran en dificultad económica, con lo cual promueven la equidad y la igualdad de oportunidades de aprendizaje entre los estudiantes, y contribuyen de forma decisiva a ser una escuela inclusiva.

Calidad

Estable, seguro y fácilmente instalable, el software libre ofrece una amplia gama de soluciones para la educación.

Para saber más

En los años 90, era realmente complicado utilizar un sistema operativo Linux y la mayoría de la cuota del mercado de los ordenadores personales estaba dominada por Windows. Encontrar drivers de Linux para el hardware que tenía tu equipo era casi una quimera dado que las principales compañías de hardware y de software no se molestaban en crear software para este sistema operativo, puesto que alimentaba la independencia de los usuarios con respecto a ellas mismas.

Afortunadamente, y gracias a la creciente presión de su comunidad de usuarios, estas situaciones pertenecen al pasado, y las compañías fabricantes de hardware han tenido que variar el rumbo. Hoy en día tenemos una gran cantidad de argumentos en los que nos podemos basar para dar el salto hacia cualquier sistema operativo basado en Linux. Tal y como podemos leer en educacionit.com, podemos encontrar las siguientes ventajas:

- Es seguro y respeta la privacidad de los usuarios: Aunque hay compañías linuxeras, como Oracle, Novell, Canonical, Red Hat o SUSE, el grueso de distribuciones y software Linux está mantenido por usuarios y colectivos sin ánimo de lucro. De esta forma, podemos confiar en que una comunidad que tiene detrás millones de usuarios, pueda validar el código fuente de cualquier de estas distribuciones, asegurándonos la calidad de las mismas, compartir posibles problemas de seguridad, y sobre todo, estar bien tranquilos con la privacidad y seguridad de nuestros datos e información personal, aspecto que debería ser crítico y determinante a la hora de trabajar con los datos de menores de edad en las escuelas y colegios.
- Es ético y socialmente responsable: La naturaleza de Linux y su filosofía de código abierto y libre hace posible que cualquier usuario con conocimientos pueda crear su propia distribución basada en otras o probar las decenas de versiones que nos podemos encontrar de una distribución Linux. Este es el caso de Ubuntu por ejemplo. Gracias a esta democratización de los sistemas operativos, incluso han podido aparecer en nuestras vidas nuevos dispositivos basados en software y hardware libre como Arduino y Raspberry Pi.
- Es personalizable: el código abierto permite su estudio, modificación y adaptación a las necesidades de los diferentes usuarios, teniendo así no un único producto sino una multiplicidad de distribuciones que satisfacen las necesidades de los diferentes colectivos a los que se dirijan. Especialmente útiles son las distribuciones educativas libres, que pueden ser adaptadas a las necesidades de las escuelas.

- Está basado en las necesidades de los usuarios y no en las de los creadores de hardware y software
- Es gratis. La mayoría de las distribuciones Linux son gratuitas y de libre descarga
- Es fácil de usar. Una de las barreras que durante años ha evitado a muchos usar Linux es su complejidad. Las distribuciones orientadas al consumo doméstico cumplen los estándares de simplicidad y necesidades que cualquier usuario sin conocimientos de tecnología pueda necesitar. El entorno gráfico es sencillo, intuitivo, e incluso se puede customizar para que se pueda parecer a los más conocidos como Windows y MacOS. Además, vienen con la mayoría de aplicaciones que cualquier usuario puede necesitar: ofimáticas, edición de audio y vídeo y navegación por Internet.
- Es suficiente. Tiene su propio market de aplicaciones. Como el resto de sistemas operativos ya sea para ordenadores o dispositivos móviles, también podemos encontrar un lugar único donde poder descargar cientos de aplicaciones para todos los gustos y necesidades.

Por estas razones, el software libre se ha expandido por toda la comunidad educativa en los últimos años de manera exponencial. Un buen ejemplo de lo que estamos hablando es **Bookstack**, este sistema de edición de contenidos para cursos que utiliza Aularagón así como el uso de **Moodle** como plataforma de enseñanza y aprendizaje. En cuanto a sistema operativo para ordenadores, en Aragón disponemos de nuestra propia distribución Linux: Vitalinux EDU. Tal y como podemos leer desde su página web: **Vitalinux EDU (DGA)** es la distribución Linux elegida por el Gobierno de Aragón para los centros educativos. Está basada en Vitalinux, que se define como un proyecto para llevar el Software Libre a personas y organizaciones facilitando al máximo su instalación, uso y mantenimiento. En concreto Vitalinux EDU (DGA) es una distribución Ubuntu (Lubuntu) personalizada para Educación, "tuneada" por los requisitos y necesidades de los propios usuarios de los centros y adaptada de forma personalizada a cada centro y a la que se ha añadido una aplicación cliente Migasfree. De ésta forma, obtenemos:

1. Un **Sistema Ligero.** Permite "revivir" equipos obsoletos y "volar" en equipos modernos. Esto garantiza la sostenibilidad de un sistema que no consume recursos de hardware innecesariamente ni obliga a la sustitución del hardware cada poco tiempo en esa espiral de obsolescencia programada en la que se ha convertido el mercado tecnológico.
2. **Facilidad en la instalación y el uso** del sistema mediante programas personalizados.
3. Un Sistema que **se adapta al centro** y/o a cada aula o espacio, y no un centro que se adapta a un Sistema Operativo.
4. **Gestión de equipo y del software de manera remota** y desatendida mediante un servidor Migasfree.
5. **Inventario** de todo el hardware y software del equipo de una forma muy cómoda.
6. Soporte y apoyo de una **comunidad** que crea, comparte e innova constantemente.

1. Gamificación

Gamificación

Con los contenidos vistos hasta ahora curso de Scratch básico, ya se ha tratado todo lo necesario para poder superar los objetivos de aprendizaje. En este módulo vamos a afianzar los conocimientos adquiridos. Para ello, vamos a crear un nuevo juego sencillo en Scratch.

Mira esta presentación, fíjate en los siguientes conceptos:

- Preguntas y respuestas
- Apuntar y pulsar
- Arcade
- Movimiento, rebote y colisiones
- Aventuras gráficas

https://www.slideshare.net/slideshow/embed_code/key/wb3kOdWlXNWNSm?startSlide=1

Scratch

Antes de nada ¿Dónde se ubica **Scracth** dentro de las estrategias de pensamiento computacional? En CATEDU hemos elaborado la siguiente hoja de ruta de toda la oferta que tenemos para el pensamiento computacional:

Importante

Para hacer este curso hay saber algunos conocimientos de (Curso básico de Aularagón por ejemplo) por si acaso, contesta estas preguntas:

https://www.youtube.com/embed/Bsr_-iWes8g

La idea del juego

Vamos a programar un juego con Scratch.

La idea de este juego es lanzar un lapicero contra una letra con el objetivo de acertar a darle. La letra se moverá para evitar ser alcanzada. Cuando lo consigamos, la letra desaparecerá, y sumaremos la puntuación de 1 punto.

Empezaremos montando los objetos que participarán en nuestro juego. Scratch tiene en su biblioteca de imágenes: letras y un lapicero. Con estas imágenes ya tenemos lo necesario para montar nuestros objetos.

Sin embargo, con idea de hacer el juego algo más original, podemos coger dibujos de Internet para montar el juego, o incluso crear con alguna herramienta de dibujo nuestros propios objetos. Para el juego que vamos a trabajar, vamos a utilizar los siguientes dibujos.
Descárgatelos a tu ordenador (botón derecho - guardar imagen como) porque vamos a montar el juego con ellos.

Creación propia, utilizando un editor de textos (OpenOffice Writer), simplemente añadiendo un rectángulo y una letra dentro, y luego capturando la pantalla.

Fuente: https://openclipart.org/detail/64429/pencil El lapicero está rotado con Gimp.

Fuente: https://openclipart.org/detail/49363/blackboard

Caso práctico: Incluir los objetos de nuestro juego

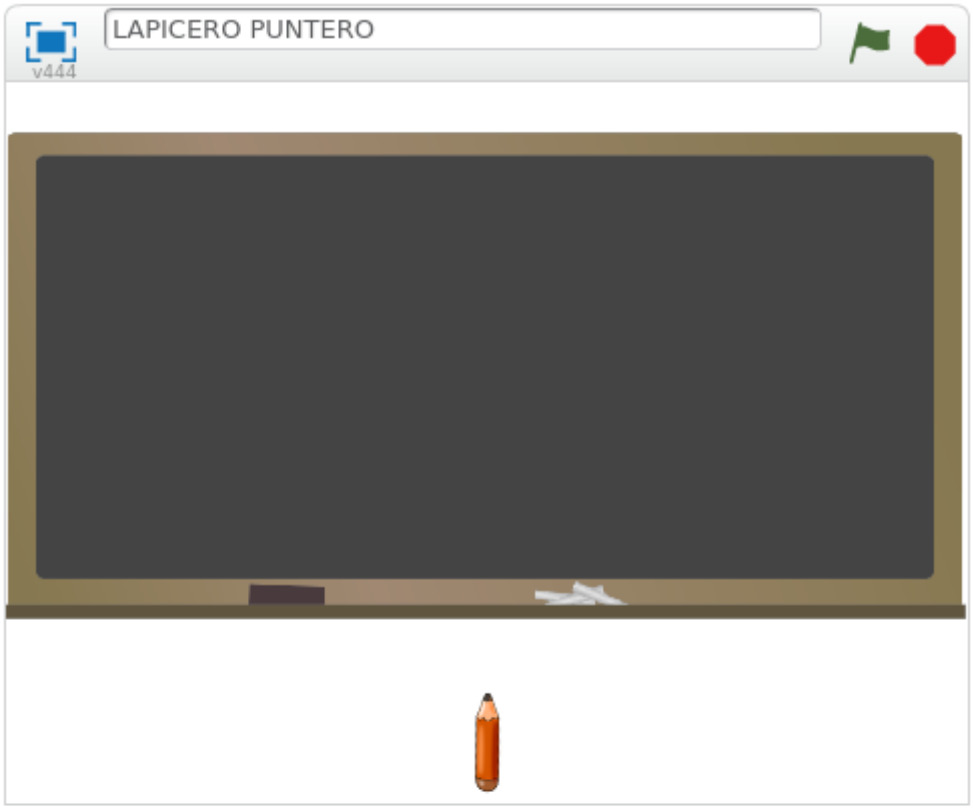
Empezamos a montar el entorno de objetos de nuestro juego.

Añade la letra A y el lápiz como objetos de Scratch. Revisa que el nombre del objeto A se llame A (botón dcho sobre el objeto - info) porque lo usaremos posteriormente.

Añade la pizarra como fondo.

Ponle un nombre al proyecto: LAPICERO PUNTERO

Solución


LAPICERO PUNTERO
v444
x: -61 y: 167
Objetos
Nuevo objeto:
Escenario
1 fondo
A
lapiz

Movimiento de la letra

Caso práctico: La letra se mueve sola

Vamos a hacer que la letra A se mueva. La letra A debe moverse de izquierda a derecha cambiando de dirección cuando llegue al borde.

1. Lo primero de todo: haz clic en el objeto de la letra A, para añadir su programa de bloques.
2. Empezaremos a arrancar las acciones al presionar Bandera (bloque Eventos: "al presionar bandera").
3. Haremos que se mueva 3 pasos (bloque Movimiento: "mover 3 pasos").
4. Cuando llegue al borde de la pantalla, haremos que cambie de dirección (bloque Movimiento: "rebotar si toca un borde").

Solución

Empezamos a construir la solución. Empezamos poniendo estos bloques:

Al arrancar el programa haciendo clic en la Bandera, veremos que la letra A se mueve 3 pasos a la derecha, pero sólo se mueve una vez.

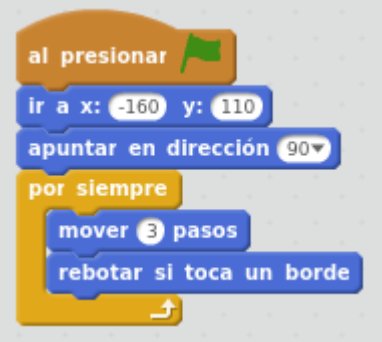
Por lo tanto, a los bloques que ya tenemos hay que **añadirles** un bloque de Control: "por siempre", y meter ahí dentro el movimiento y control de rebote. Esto genera un bucle, de forma que la letra A no deje de moverse nunca.

Siguente paso: a los bloques que ya tenemos, les metemos el bloque "por siempre", y el programa quedará de la siguiente forma:

Para asegurarnos que la letra A está posicionada siempre en la misma posición al iniciar el juego, ponemos su posición fija al inicio, por ejemplo con con x=-160 y=110 (bloque Movimiento: "ir a x: -160 y: 110").

Hacemos que A apunte a la derecha para que empiece a moverse hacia la derecha (bloque Movimiento: "apuntar en direccion 90"). Es posible que el programa funcione bien incluso sin este bloque, pero así nos aseguramos que la letra se va a empezar a mover hacia la derecha y no hacia otro lado.

Con esto ya hemos terminado el PROGRAMA que hace que la letra se mueva sola, y siempre empiece desde la misma posición.



Movimiento del lápiz

Caso práctico: mover el lápiz

Haremos que el lápiz se mueva a izquierda y derecha al presionar flechas izquierda y derecha.

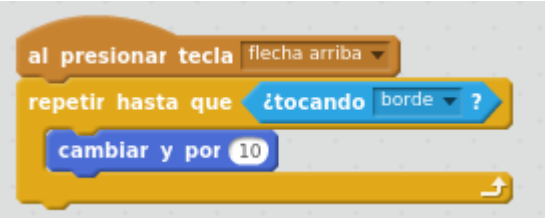
Solución

Caso práctico: lanzar el lápiz

Al presionar Flecha arriba, haremos que el lápiz se mueva constantemente hacia arriba, hasta llegar al borde de arriba.

- Iniciamos el programa con el bloque de Control "al presionar tecla flecha arriba".
- Utilizar el bloque Movimiento "cambiar y por 10" para hacer que el lápiz se mueva hacia arriba.
- Añadir el bloque de Control "repetir hasta que < >" para hacer que el bucle se repita constantemente, y que se salga del bucle al llegar a una condición. Habrá que meter el bloque de Movimiento dentro del bucle.
- Utilizar el bloque Sensor "tocando.. borde" para detectar que hemos llegado hasta arriba. Esta será la condición de salida del bucle.

Solución



Cuando llegue el lápiz arriba del todo, es decir: cuando se termine el bucle que comprueba si el lápiz ha tocado un borde, lo volvemos a poner en su situación inicial.

Solución

Colisión del lápiz con la letra

Caso práctico: detectar colisión con la letra

El objetivo es alcanzar con el lápiz la letra que se mueve.

Para controlar la colisión seguimos añadiendo bloques en la programación del objeto lápiz, en el programa que se inicia "al presionar tecla flecha arriba".

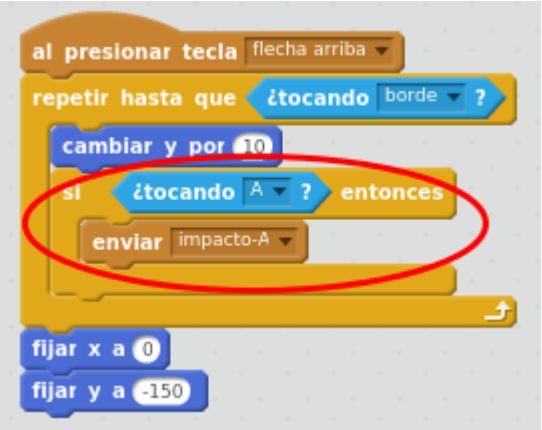
Si detectamos que estamos tocando la letra A, enviaremos el mensaje "impacto-A" a la letra A para decirle que desaparezca.

Habrá que pensar dónde colocar estos bloques.

Solución

Estos son los bloques que controlarán que si el lápiz toca el objeto A, entonces se envía el mensaje "impacto-A":

Estos bloques los colocaremos justo después de haber movido el lápiz en 10 posiciones hacia arriba:



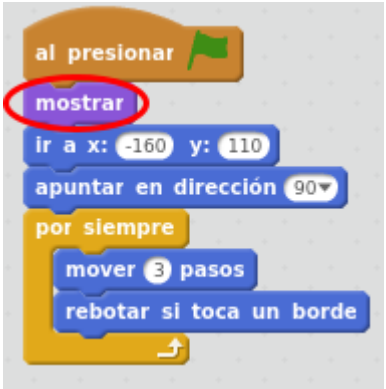
Ahora tenemos que recoger el mensaje "impacto-A" en el objeto A. Por lo tanto, haremos clic en el objeto A para modificar los bloques de su programación.

Añadiremos un bloque de programación en el objeto A: Cuando se reciba el mensaje "impacto-A", haremos que el objeto A desaparezca.

Solución

Ahora, cuando el objeto A es impactada por el lápiz, desaparece y ya no vuelve a aparecer. Tendremos que hacer que al inicio del juego, el objeto A aparezca visible. Lo podremos añadir en el programa ya existente que mueve la letra A.

Solución

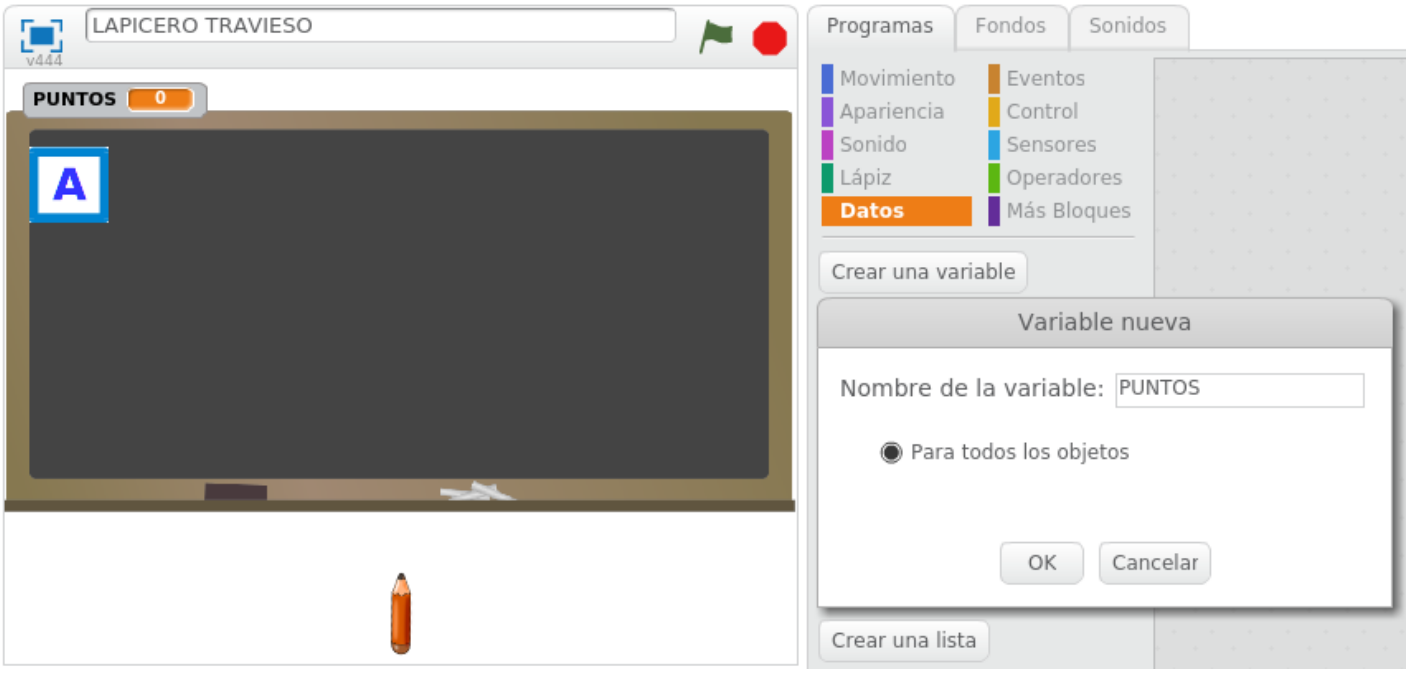


Caso práctico: Añadir puntuación

Añadiremos un marcador de puntos al juego. Haremos que los puntos se incrementen al detectar la colisión del lápiz con la letra.

Crear una variable llamada PUNTOS.

Solución



En la programación del objeto A:

- Al inicio del programa, ponemos los puntos a 0.
- En el momento de detectar colisión, sumamos 1 en la variable puntos.

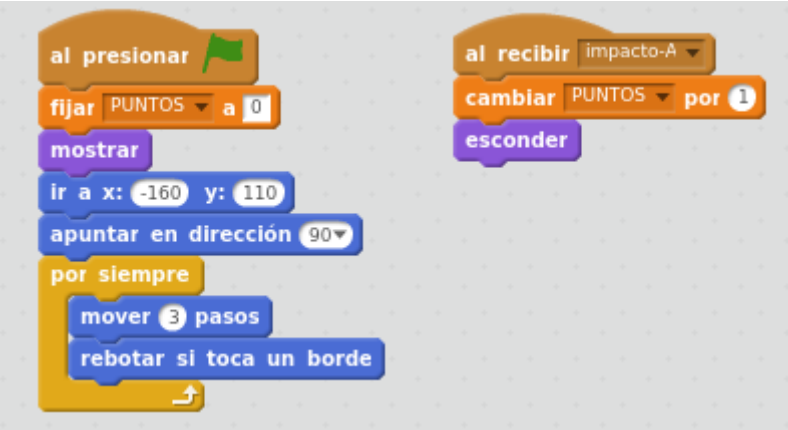
Solución


al presionar
fijar PUNTOS a 0
mostrar
ir a x: -160 y: 110
apuntar en dirección 90
por siempre
mover 3 pasos
rebotar si toca un borde
al recibir impacto-A
cambiar PUNTOS por 1
esconder

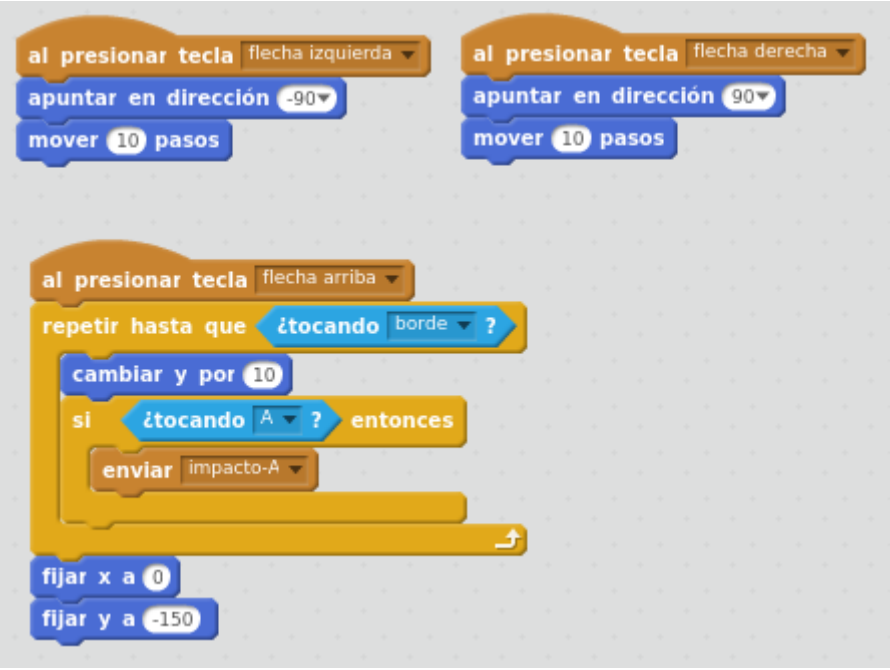
Resultado final

Hasta aquí, hemos creado una programación para el juego de lanzar un lápiz contra una letra.

Los programas del objeto A queda por tanto de la siguiente forma:



Los programas del objeto lapiz quedan por tanto de la siguiente forma:



¡YA LO TENEMOS!. Utilizando Pensamiento Computacional, hemos conseguido montar un pequeño juego con Scratch.

2. Gamificación (continuación)

Gamificación (continuación)

Es este bloque de contenidos se plantea un reto mayor: continuar mejorando el juego que ya tenemos creado. Va a requerir añadir más complejidad en los programas de los objetos, ¡pero nada que no esté ya a vuestro alcance! Espero que os sintáis con ganas de continuar con este bloque de contenidos que se sale de la programación planteada en el curso de Aularagon.

El resultado de lo que vamos a conseguir lo puedes ver aquí:

Ver resultado final

https://scratch.mit.edu/projects/embed/125282917/?autostart=false

La idea del juego ampliada

Vamos a programar un nuevo juego con Scratch, que va a ser una ampliación del anterior. La idea de este juego es la del típico matamarcianos. Pero en nuestro caso, lo que vamos a hacer es disparar pintura con un lápiz a unas letras. Las letras se moverán para evitar ser alcanzadas. El objetivo final será acertar a disparar a todas las letras.

Empezaremos montando los objetos que participarán en nuestro juego. Scratch tiene en su biblioteca de imágenes: letras, lapicero, rayo. Con estas tres imágenes ya tenemos lo necesario para montar nuestros objetos.

Sin embargo, con idea de hacer el juego algo más original, podemos coger dibujos de Internet para montar el juego, o incluso crear con alguna herramienta de dibujo nuestros propios objetos. Para el juego que vamos a trabajar, vamos a utilizar los siguientes dibujos. Descárgatelos a tu ordenador (botón derecho - guardar imagen como) porque vamos a montar el juego con ellos.

Creación propia, utilizando un editor de textos (OpenOffice Writer), simplemente añadiendo un rectángulo y una letra dentro, y luego capturando la pantalla.

Creación propia, utilizando un editor de textos (OpenOffice Writer), simplemente añadiendo un rectángulo y una letra dentro, y luego capturando la pantalla.

Creación propia, utilizando un editor de textos (OpenOffice Writer), simplemente añadiendo un rectángulo y una letra dentro, y luego capturando la pantalla.


Fuente: https://openclipart.org/detail/29133/pencil El lapicero está rotado con Gimp.


Creación propia, utilizando un programa de edición de imágenes (Gimp)

Fuente: https://openclipart.org/detail/49363/blackboard

Caso práctico: incluir los objetos de nuestro juego

Empezamos a montar el entorno de objetos de nuestro juego.

Añade las letras A, B y C, el lápiz y el disparo como objetos de Scratch. Revisa que los nombres de objetos de las letras A, B y C se llamen a, b, y c, (botón dcho sobre el objeto - info) porque usaremos estos nombres posteriormente.

Añade la pizarra como fondo.

Ponle un nombre al proyecto: LAPICERO TRAVIESO

Solución

Te tienes que quedar así:

x: 240 y: -176

Objetos

Nuevo objeto:

Escenario
1 fondo

Fondo nuevo:

a

b

c

lapiz

disparo

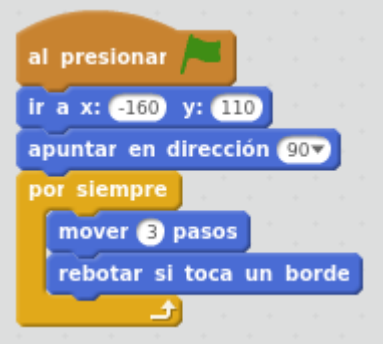
Movimiento de la letra

Caso práctico

Vamos a hacer que la letra A se mueva. La letra A debe moverse de izquierda a derecha cambiando de dirección cuando llegue al borde. Esto ya lo hemos hecho en el juego anterior.

Al presionar bandera, posicionamos la letra en un lugar fijo y apuntando a la derecha. A continuación haremos que, de forma indefinida, la letra se mueva 3 pasos, y cuando llegue al borde de la pantalla, haremos que cambie de dirección.

Solución



Movimiento del lápiz

Caso práctico: el lápiz se mueve con las teclas izquierda y derecha

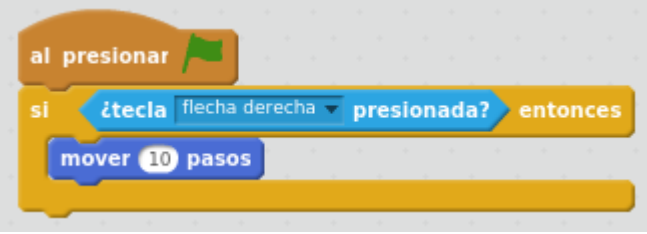
Queremos que el lápiz se mueva a la izquierda cuando presionemos la flecha izquierda. Y que se mueva a la derecha cuando presionemos la flecha derecha.

Esto ya lo hemos hecho en el juego anterior. Sin embargo, si somos "exquisitos", habrás observado que si mantienes pulsada una tecla de flecha izquierda o derecha, debido a que hay un retardo en el teclado antes de empezar a mandar la señal de repetición a nuestro programa, se produce también un retardo en el primer movimiento del lápiz.

Vamos a construir una solución que evita este inconveniente con la siguiente lógica: Al presionar Bandera, estaremos comprobando de forma indefinida si hay alguna tecla presionada. Moveremos el lápiz 10 pasos a la derecha si se presiona la tecla "flecha derecha", y moveremos el lápiz 10 pasos a la izquierda si se presiona la tecla "flecha izquierda".

Empezamos a construir la solución paso a paso. En primer lugar, vamos a hacer que al presionar Bandera, haremos que si la tecla "flecha derecha" está presionada, mover el lápiz 10 pasos a la derecha

Solución



Pero esto no funciona. porque estos bloques sólo se arrancan una vez. Y necesitamos que el programa esté constantemente comprobando si la tecla está presionada. Por lo tanto, tenemos que poner todo dentro el bloque de Control "por siempre", para que el programa esté constantemente comprobando si la tecla está presionada. Ahora ya habremos conseguido que el lápiz se mueva a la derecha cuando presionemos la tecla de flecha derecha,

Solución

Lo único que queda es añadir la comprobación de si la tecla de flecha izquierda esté presionada, se mueva el lápiz 10 pasos a la izquierda.

Solución

Movimiento del disparo

Caso práctico: disparar al presionar espacio

Vamos a hacer que cuando se presione la tecla espacio, el disparo de pintura salga del lápiz y empiece a moverse hacia arriba hasta que llegue al borde superior.

Empezamos haciendo que el disparo se mueva en 20 pasos hacia arriba: Fijaremos la posición Y del objeto disparo sumándole 20 a su posición actual. Utilizaremos el bloque de Operadores " + "

Solución

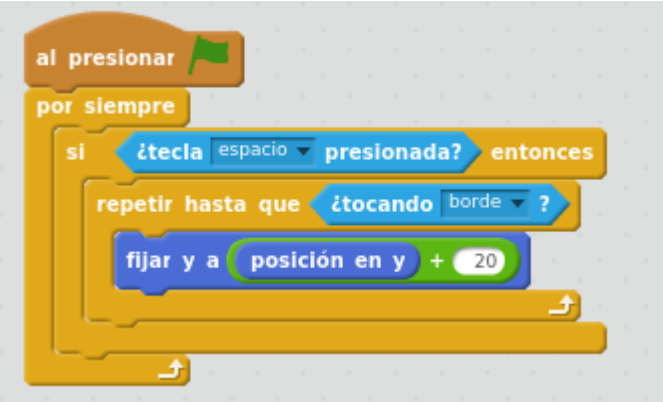
Ahora, el disparo tiene que moverse 20 pasos hacia arriba pero constantemente, hasta que llegue al borde.

Solución

Ponemos como condición previa a todo esto que se ponga en marcha si se ha presionado la tecla espacio.

Solución

Esto sólo se arranca una vez, necesitamos meter un bucle para que el programa esté constantemente comprobando si se ha presionado la tecla espacio:



El disparo debe salir del lápiz, por tanto, debemos posicionar el disparo encima del lapicero. posicionaremos el disparo 50 pixeles por encima de la posición del lápiz. ¿En qué momento hay que posicionarlo?: antes del bucle que mueve el disparo hacia arriba.

Solución

¿Cuándo mostrar y ocultar el disparo?

- Al principio del todo, debemos ocultar el disparo.
- Una vez que tenemos el disparo posicionado encima del lapicero, lo mostramos

- En última instancia, hacemos que desaparezca después de detectar que ha tocado el borde.

Solución

Colisión del disparo con un objeto

Caso práctico: control de la colisión del disparo con una letra

Prueba

La colisión la vamos a controlar dentro del movimiento del disparo, por lo que continuaremos añadiendo más programación al objeto de disparo. También necesitaremos añadir más programación en las letras. El objetivo es que mientras se está moviendo el disparo, comprobar si colisiona con alguno de los objetos (a, b, c). Si colisiona, quitaremos la letra de la escena, para que no moleste, y sumaremos ¡1 punto al marcador!

Dentro del objeto disparo, si el objeto disparo toca al objeto A, enviaremos el mensaje "impacto-A". Este mensaje habrá que crearlo en el bloque Eventos - "enviar..." - nuevo mensaje.

Solución

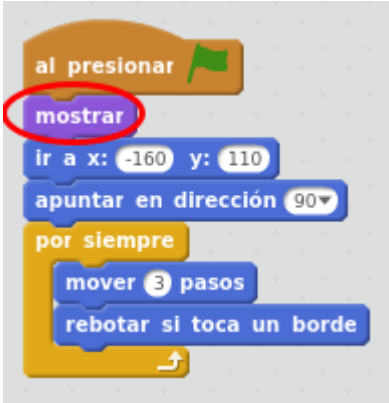
¿Dónde colocaremos estos bloques de programación?: en la programación del objeto disparo, justo después de haberlo movido 20 pasos hacia arriba:

En la programación del objeto A, necesitaremos recoger el mensaje "impacto-A" en el objeto A. Una vez recogido el mensaje, ocultaremos el objeto A.

Solución

Ahora resulta que cuando el objeto A es impactado, se oculta y ya no se muestra ni aunque iniciemos una nueva partida. La solución es hacer que al inicio del juego, el objeto A se muestre.

Solución

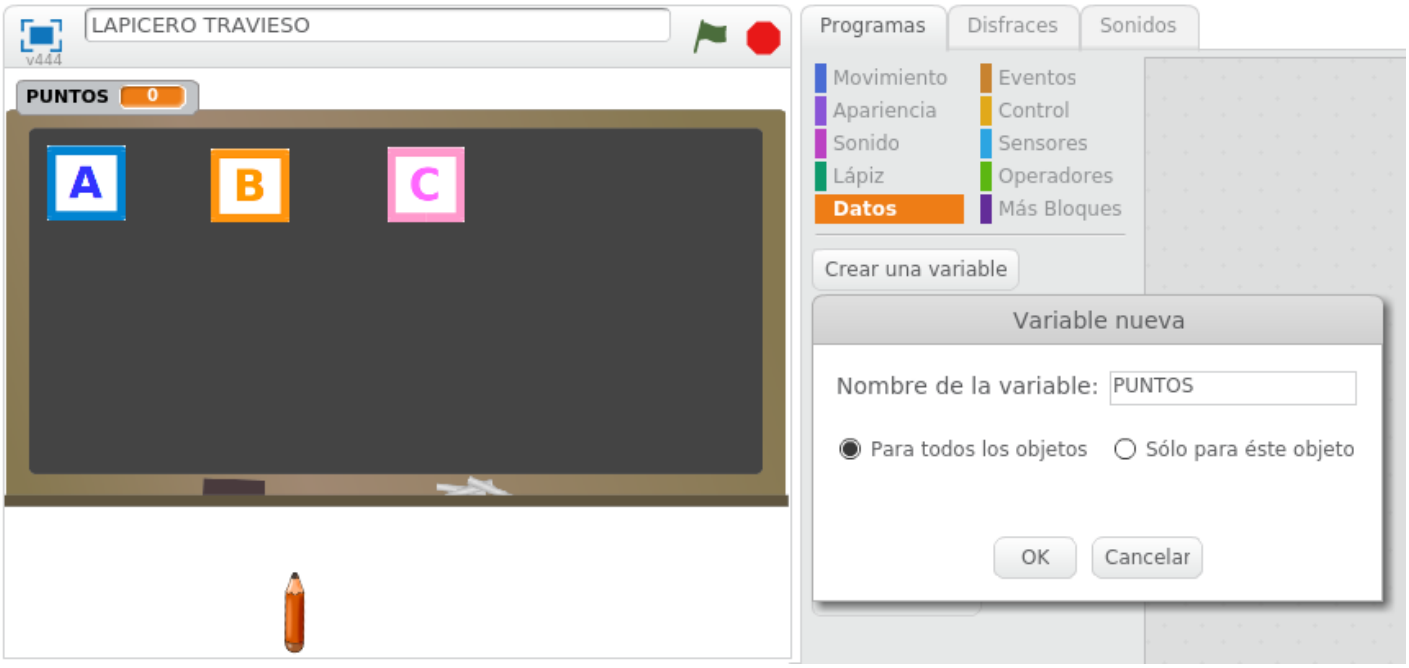


Caso práctico: ¡Ahora hay que sumar los puntos del juego!

Tendremos los puntos de juego visibles en la pantalla. Haremos que al colisionar el disparo con un objeto, se sume 1 punto.

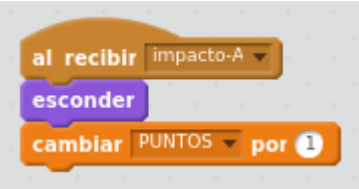
Hay que añadir una variable de puntos. Al añadirla, la variable se verá en el escenario de juego, la posicionaremos en la zona superior izquierda.

Solución



En la programación del objeto A, cada vez que se detecte que el objeto A ha sido colisionado por el disparo, se sumará 1 punto en la puntuación del juego.

Solución



Últimos ajustes

Caso práctico: Programar las letras B y C

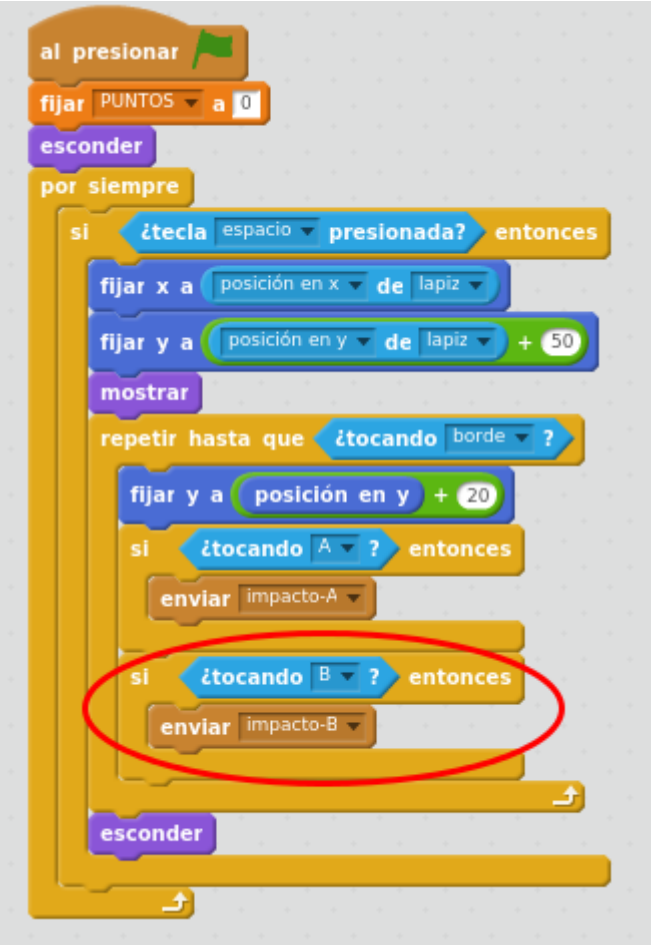
Ahora hay que programar las letras B y C igual que la A para que tenga el mismo comportamiento. Su programación será prácticamente igual a la del objeto A, cambiando el nombre del mensaje y sus coordenadas.

- Para no empezar de cero, es posible arrastrar todos los bloques de programación de la A sobre el objeto B.
- Posteriormente, hay que cambiar datos específicos para objeto B:
- El mensaje será "impacto-B"
- Las coordenadas iniciales para B serán x=0 y=110

De igual forma a lo realizado en la letra B, replicamos la programación para el objeto C.

- Su mensaje será "impacto-C"
- La posición fija del objeto C puede ser x=160 y=110

Solución

OTRA OPCIÓN: Podríamos duplicar el objeto A. Posteriormente habría que:

- Añadirle el disfraz de letra B
- Borrarle al objeto el disfraz de letra A.
- Cambiarle el nombre al objeto por "A" (Botón dcho del ratón sobre el objeto B - info)
- Cambiar los datos especiíficos del objeto B (mensaje "impacto-B", coordenadas -30, 110), y no olvidar añadir la comprobación de colisión en la programación del disparo.

Caso práctico: Añadir un temporizador

Añadamos un temporizador a nuestro juego.

Crear una variable llamada TIEMPO.

En la programación del Fondo (los fondos también pueden tener sus propios programas), incluiremos un nuevo programa. Al inicio reiniciaremos el cronómetro. En un bucle que se ejecutará siempre, pondremos el valor del cronómetro en la variable TIEMPO y así se visualizará en pantalla.

Solución

Caso práctico: último reto, controlar el fin de juego

Cuando hayamos impactado con todas las letras, mostrar un mensaje en pantalla informando de "Fin de juego".

Una opción sería controlar constantemente el valor de los puntos, y hacer que cuando lleguen a 3, se acabe el juego. Pero otra posible solución algo mejor pensada puede ser controlar en una lista las letras que han sido impactadas. Para ello:

En la programación del lapicero, añadir un nuevo programa, que al iniciar el juego esté comprobando constantemente si hemos impactado las tres letras, y en tal caso mostraremos el mensaje "Fin de juego". Cómo podemos hacerlo:

- Añadiremos un disfraz nuevo al objeto lapiz. Construimos nosotros mismos el disfraz que va a ser el texto "Fin de juego".
- Crear una lista (pestaña Programas - Datos - Crear una lista, que podemos llamar LETRAS).
- Añadimos un bloque de programa en el objeto lapiz, que se inicie al presionar Bandera, donde de forma indefinida comprobaremos si la lista de letras contiene "a", "b" y "c". En tal caso, cambiaremos el disfraz del lapiz por el disfraz de "findejuego" y detendremos todos los programas.
- Tendremos que hacer que al inicio del juego, el disfraz del lapiz sea el del lapicero, para que no se quede fijo el mensaje de "Fin de juego". Y también habrá que ocultar y vaciar la lista.

En cada objeto de letra: añadir un valor a la lista cada vez que se impacte una letra.

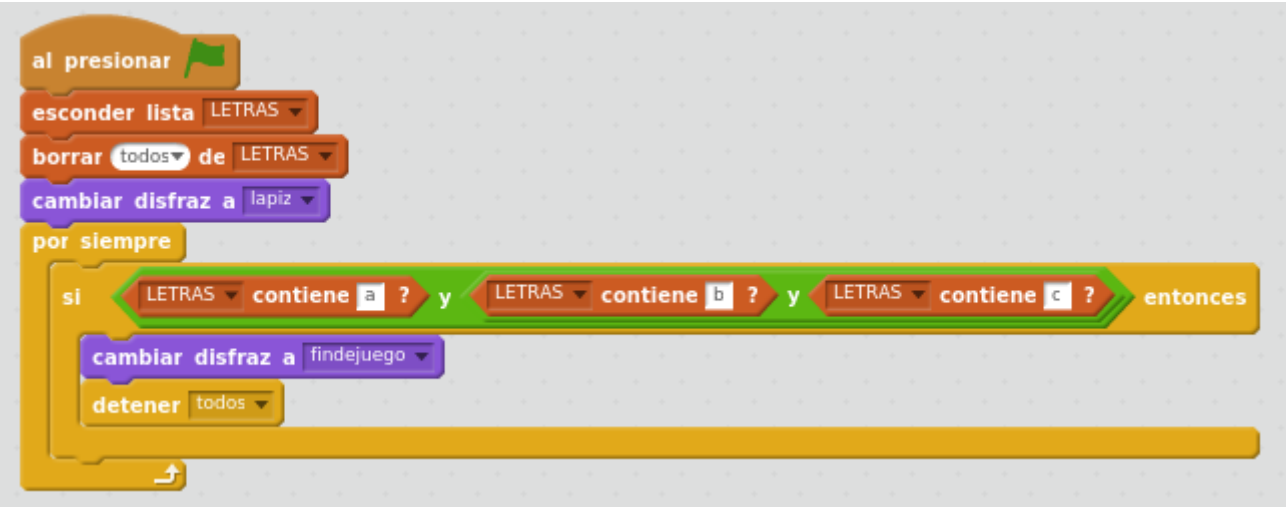
En el objeto de lapiz: Crear una lista. Ocultarla y vaciarla al inicio del juego.

Solución

Crear una lista (pestaña Programas - Datos) y la llamamos LETRAS.

En la programación de las letras, añadir este bloque tras recibir el mensaje de impacto:

El programa del lápiz para controlar el fin de juego quedaría así:

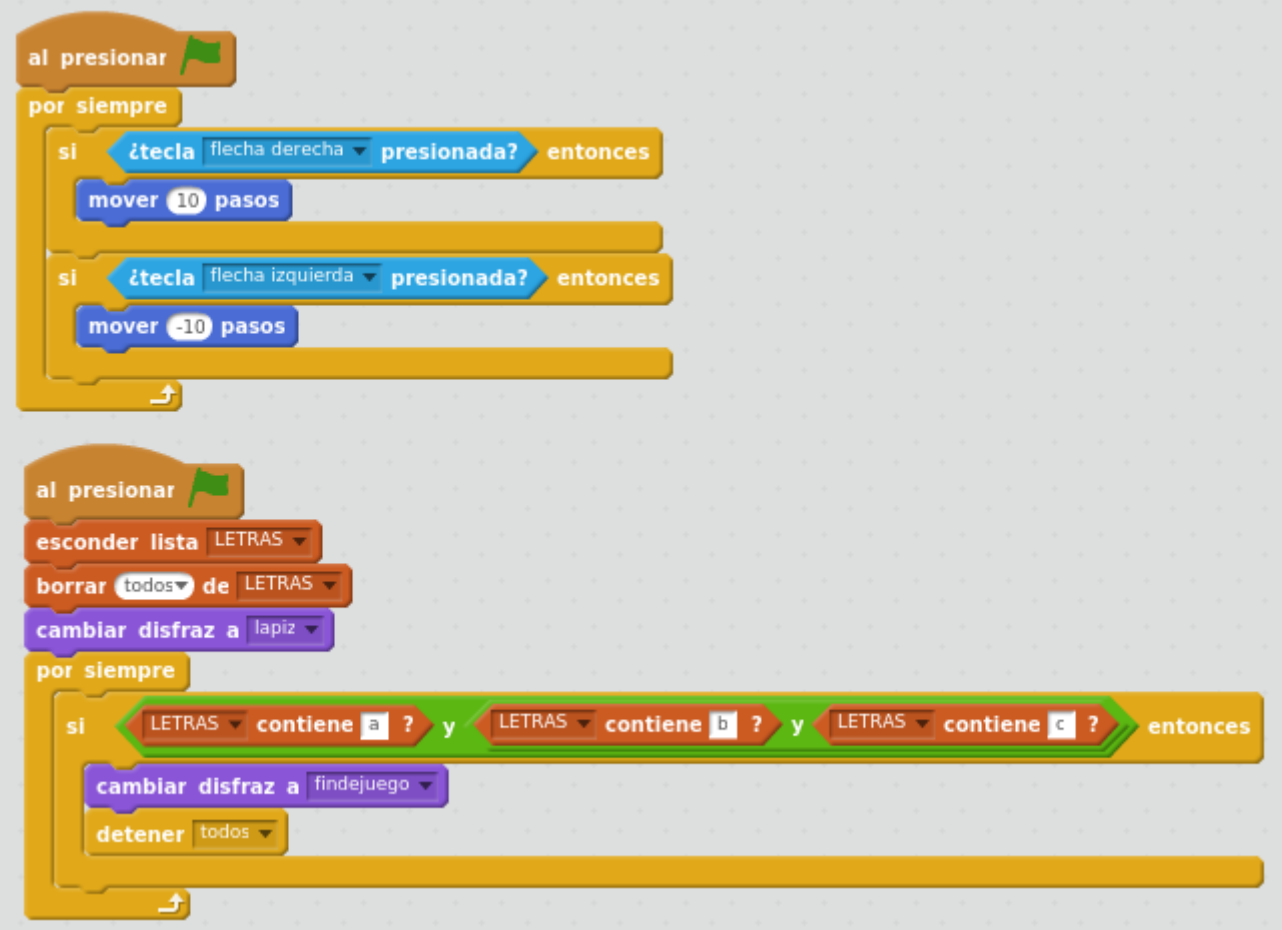


Resultado final

Hemos creado una programación para el juego de lanzar pintura desde un lápiz contra varias letras.

Los programas de cada letra, por ejemplo de la letra A, quedan de la siguiente forma:

Los programas del lápiz quedan:



El programa del disparo queda:

El programa de la imagen de fondo queda:

¡YA LO TENEMOS!. Utilizando Pensamiento Computacional, hemos conseguido montar un juego completo con Scratch.

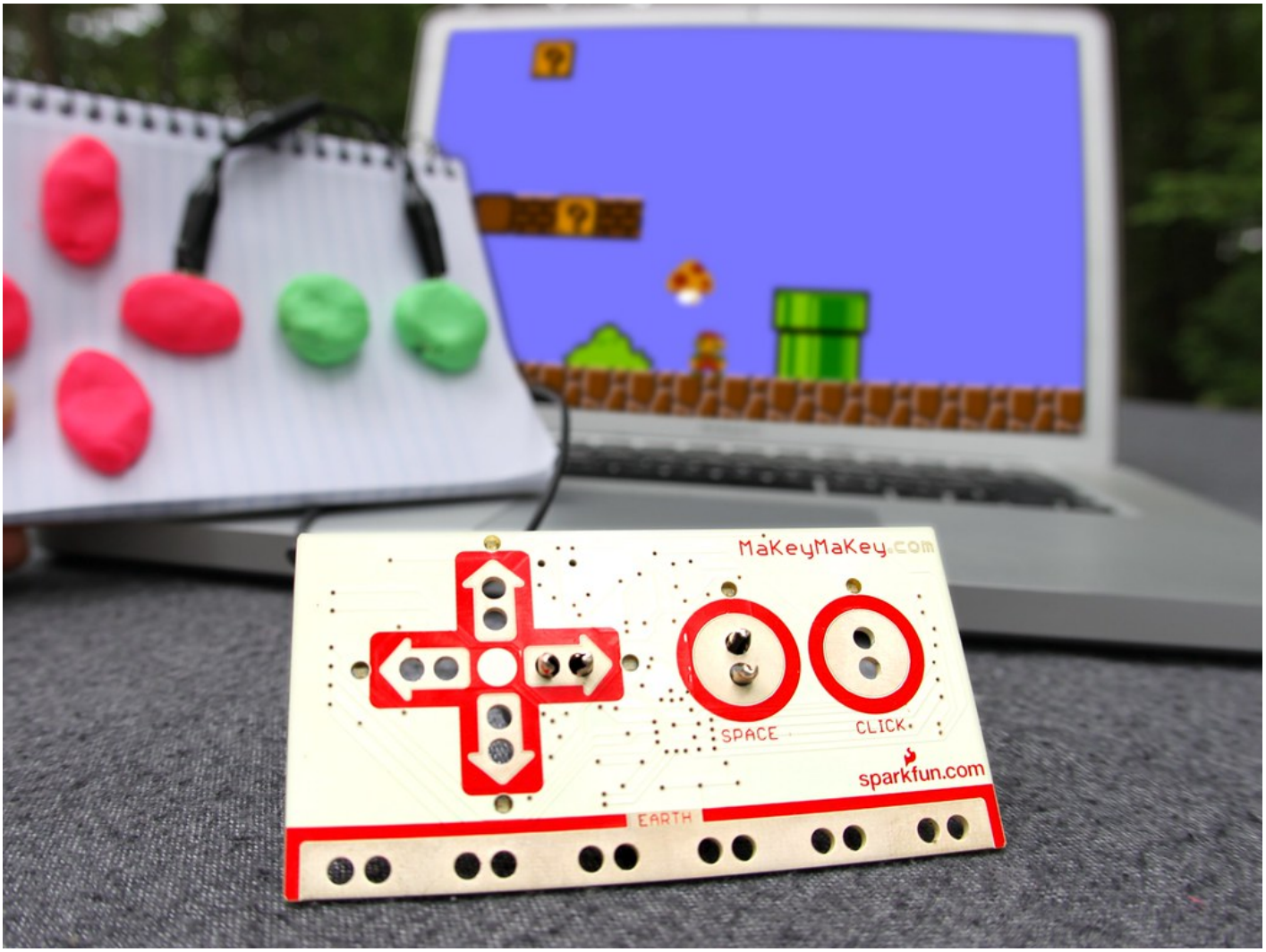
Ver resultado final

https://scratch.mit.edu/projects/embed/125282917/?autostart=false

3. Abriendo posibilidades

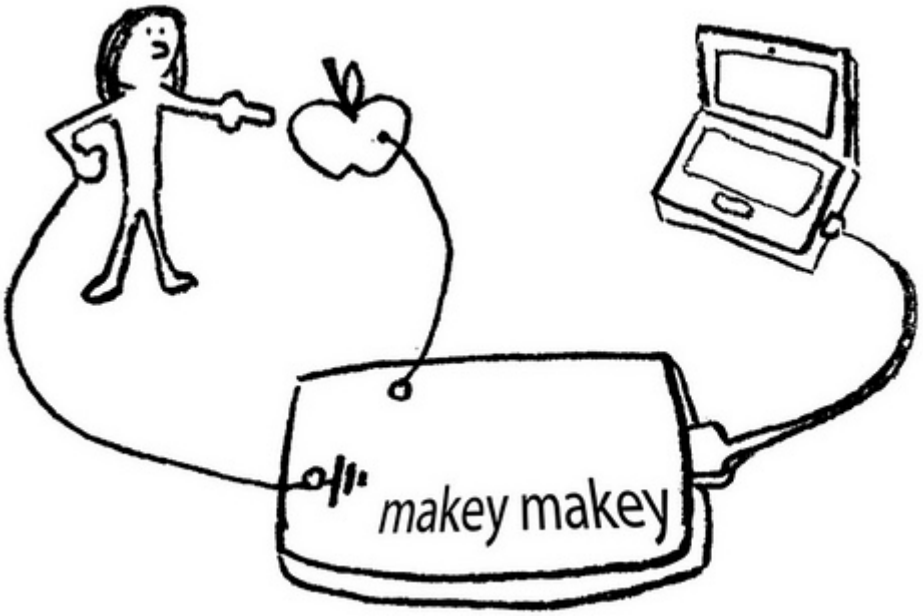
Abriendo posibilidades

En este módulo, algunas secciones sólo funcionan con Scratch on-line, evidentemente las que son de Internet.




forever
imagine
program
share

Makey Makey



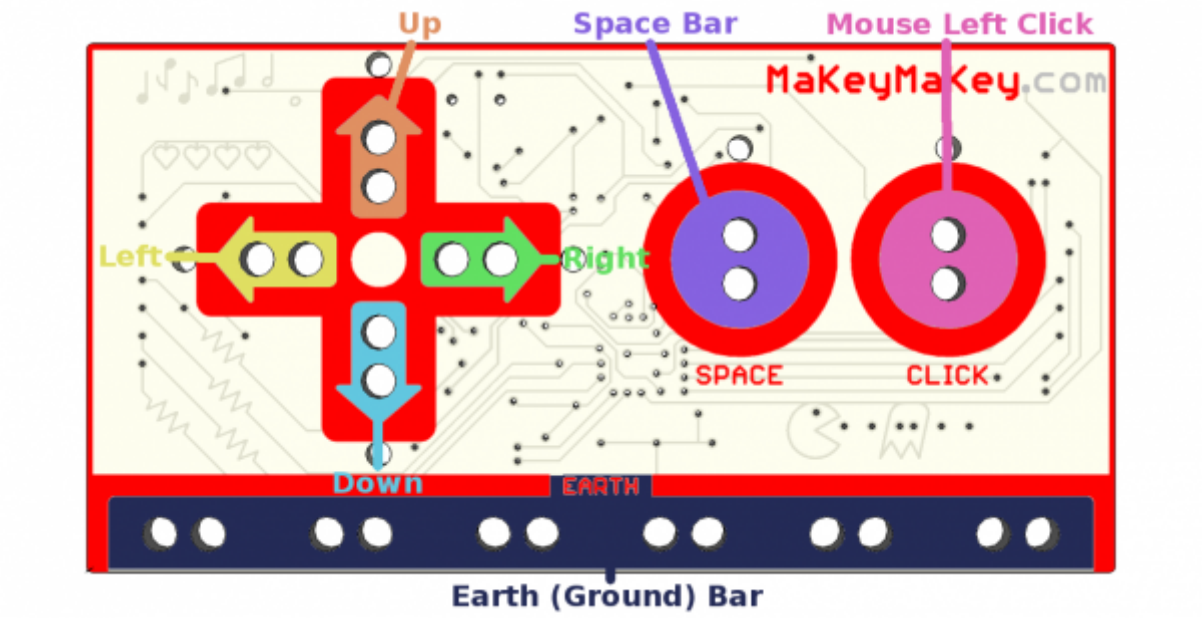
Puedes pedir un Makey Makey para hacer este curso en CATEDU-ROBOTICA

¿Qué es?

Es simplemente una entrada al ordenador por USB, como un teclado o ratón, pero por contacto:

https://www.youtube.com/embed/rfQqh7iCcOU

Las entradas principales son las flechas del teclado, la barra espaciadora y el click del ratón

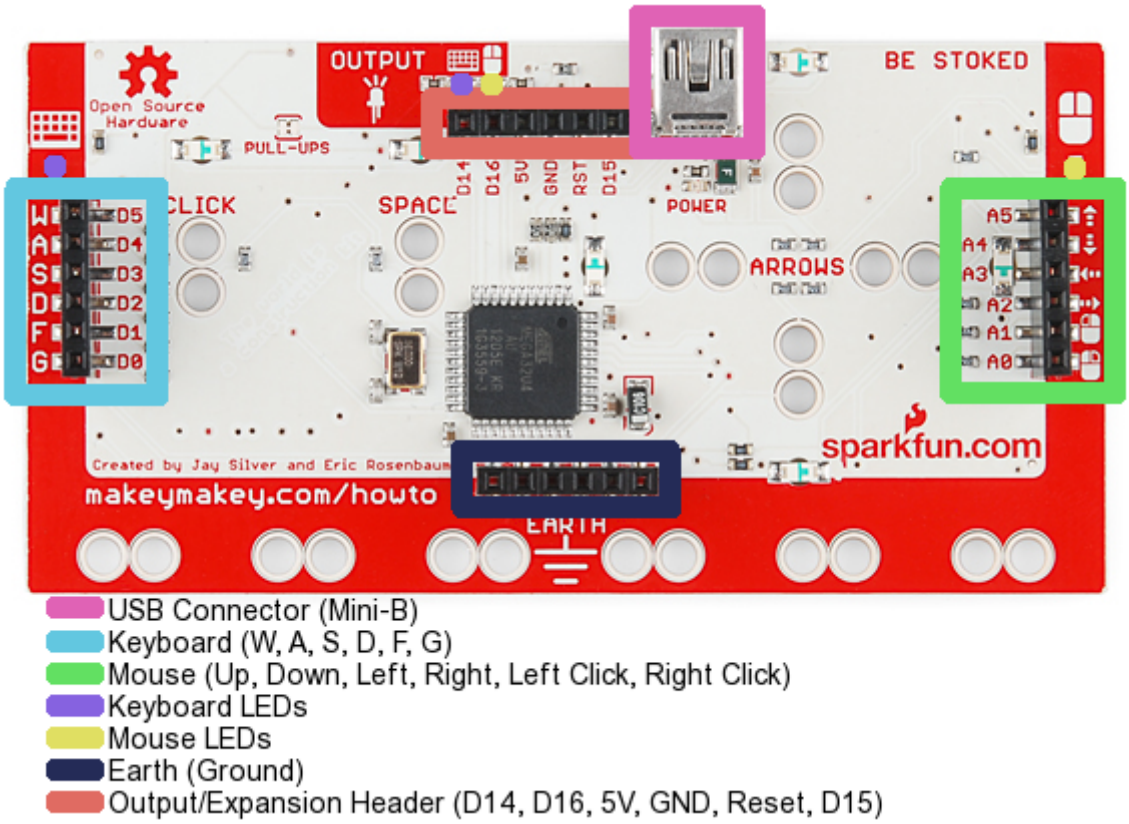


Fuente: https://learn.sparkfun.com/tutorials/makey-makey-quickstart-guide

Debajo de la placa tiene acceso a

- las teclas **W, A, S, D, F, y G**
- **botón derecho, izquierdo del ratón** y sus **movimientos**

1. **Conectas el Makey Makey por USB en tu ordenador**
2. **Conectas los plátanos a los pines WASDF (ver figura de abajo)**
3. **Ejecuta un programa Makey Makey que diga un tono cuando se pulsa una de las teclas WASDF**
4. **A disfrutar !**



Fuente: https://learn.sparkfun.com/tutorials/makey-makey-quickstart-guide

Se pueden alterar estos parámetros entrando en su código y modificandolo [+info]

Cuesta alrededor de 70€

Plastilina I

Propuesta Cuadro Musical

Puede ser un cuadro, o también un mapa, un panel gráfico con las partes del cuerpo, relieve ...

https://www.youtube.com/embed/CB_O7GFDZBc

Solución

Para hacer la plastilina conductora aquí tienes una receta:

http://www.comofuncionainternet.net/circuitos-con-plastilina/

Una pega: No dura nada, a los pocos días se endurece o se pudre y huele mal.

Conectas la tecla que quieras con la música elegida, en este caso conecto la plastilina con el **"a"** del Makey y que suene Vivaldi

Otras propuestas con plastilina

Piano

https://www.youtube.com/embed/iar9oJJj4Lc

Conductor o no conductor

https://www.youtube.com/embed/OHk2YBEtmDY

Plastilina II

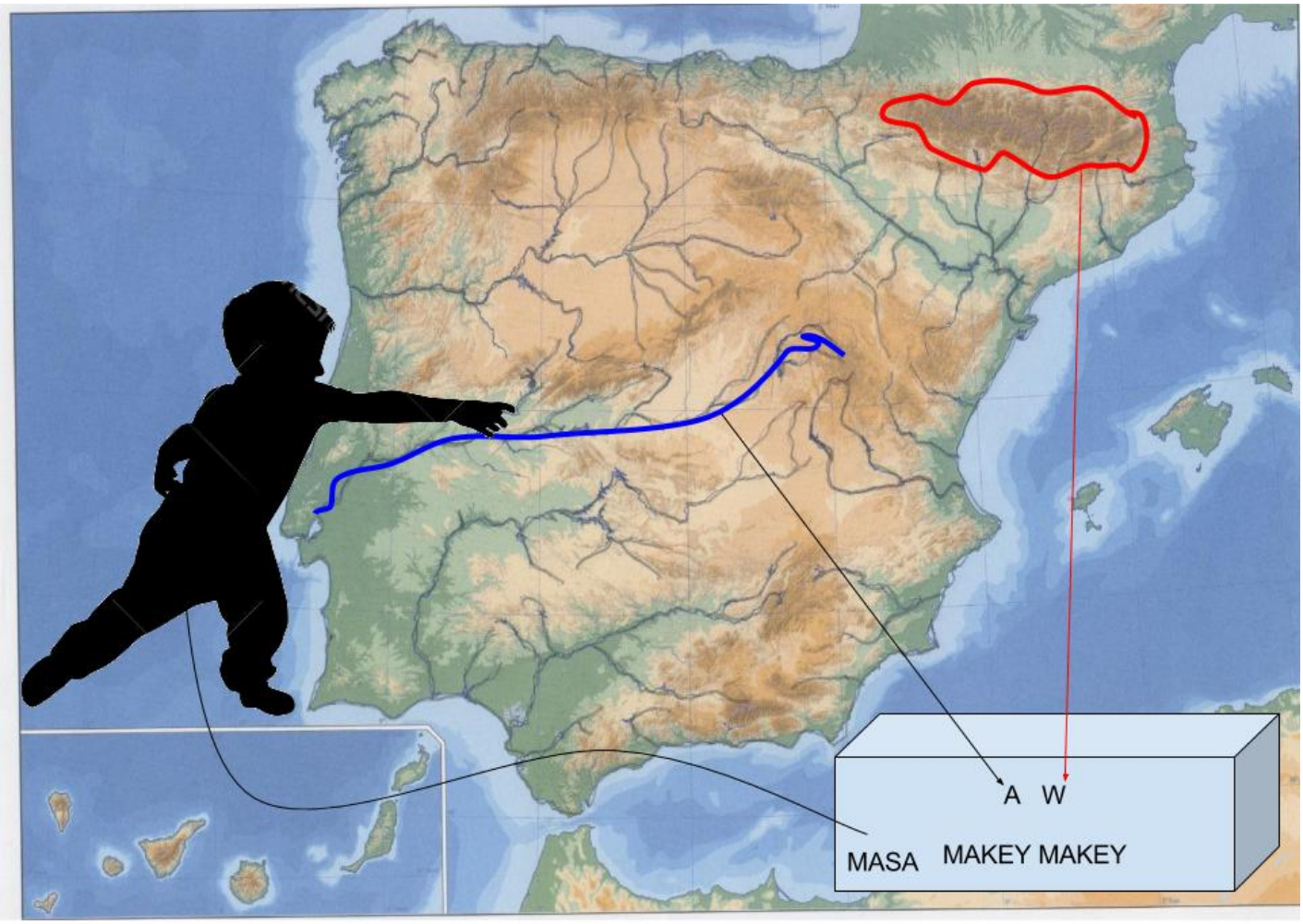
Propuesta

Hacer un mapa de relieve interactivo con plastilina

Por ejemplo: Ponemos el mapa físico de la península y plastilina conductora:

- Una plastilina en **Pirineos** lo conectamos al **W** del Makey Makey
- Otra plastilina en el rio Tajo lo conectamos al **A** del Makey Makey
- el niño tiene que tocar la masa del Makey Makey

Hacer un programa en Scratch que vaya preguntando y si contestas bien que salga aplausos y un texto "bien" en caso contrario un sonido de fallo y el mensaje "No no"



Solución

Este es el proyecto: https://scratch.mit.edu/projects/124217131/

si no tienes el Makey Makey no pasa nada, pulsa con el teclado A y W y a jugar !!!

https://scratch.mit.edu/projects/embed/124217131/?autostart=false

Piano

Propuesta

Hacer un piano humano

https://www.youtube.com/embed/M-8-JaDIgY0%20

Solución

Primero tenemos que definir qué teclas son qué notas, nuestra propuesta es la siguiente:

Tecla	Nota
Flecha izq	48
Flecha abajo	50
Flecha arriba	52
Flecha der	53
w	55
a	57
s	59
d	60

El programa en sí es muy básico:

Puedes probarlo aquí pero utilizando las teclas del teclado y del ratón

https://scratch.mit.edu/projects/embed/123624393/?autostart=false

La única dificultad es la construcción: Cada tecla del Makey Makey se conecta con un cable, y al

otro extremo una persona que hace de esa tecla.

El que toca el piano tiene que estar conectado a la masa del Makey Makey con un cable

Los niños se lo pasan genial:

https://www.youtube.com/embed/jv2vGhF0cV8

O no tan niños ;)

https://www.youtube.com/embed/M-8-JaDIgY0%20

ALTERNATIVAS

Con bananas, plastilina, gominolas....

https://www.youtube.com/embed/gyBRvFvs3Mk

El cuerpo humano

Propuesta

Esta propuesta es de Programo Ergo Sum, http://www.programoergosum.com/ agradecemos su colaboración

https://www.youtube.com/embed/SjsuJ_ozv9E

Solución

https://www.youtube.com/embed/MD-aHaoXMow

https://www.youtube.com/embed/m791U-d_qYk

Cuelga experiencias

En este muro puedes colgar experiencias interesantes tuyas o que veas por internet

https://padlet.com/CATEDU/makey

https://padlet.com/embed/phc0rpzhe1qj

ProgramoErgoSum

Esta sección la queremos agradecer al autor de la página http://www.programoergosum.com/ que nos ha autorizado publicar sus vídeos.

Contínuamente el autor sube propuestas, recomendamos visitar su canal de vídeo Youtube y suscribirse para estar al día.

Vocales

Propuesta

https://www.youtube.com/embed/01DvtPs0x4M

Solución

https://www.youtube.com/embed/WXUuqY-2yY4

https://www.youtube.com/embed/PgrZqwEnPNA

https://www.youtube.com/embed/-I3CAcOF1s0

Clonar Matar pájaros

Propuesta

Esta es una buena propuesta de PROGRAMO ERGO SUM - CLONAR

https://www.youtube.com/embed/E74_nUAaZbY

%accordion%Solución%accordion%

Preliminares

https://www.youtube.com/embed/E74_nUAaZbY

Pájaro azul: clonar, movimiento, punto de mira, vidas ...

https://www.youtube.com/embed/N2k0wWMiDv4

Ahora el pájaro blanco:

https://www.youtube.com/embed/Heo9pkR9iqc

2 jugadores Fútbol

Propuesta

Nuestra propuesta es de Programo Ergo sum - futbol

https://www.youtube.com/embed/n5JEJ8NV7V4

Solución

https://www.youtube.com/embed/sUC0JvL1rOY

https://www.youtube.com/embed/aoksH0RSgO0

https://www.youtube.com/embed/nKdrurkjr2I

Bloques Duck Hunt

Propuesta

La mejor forma de aprender los bloques es con un ejemplo de Programo Ergo Sum:

Solución

Preliminar

https://www.youtube.com/embed/wUIJIWI6WsU

Aquí es donde se emplea los bloques

https://www.youtube.com/embed/XUFgF33qhX0

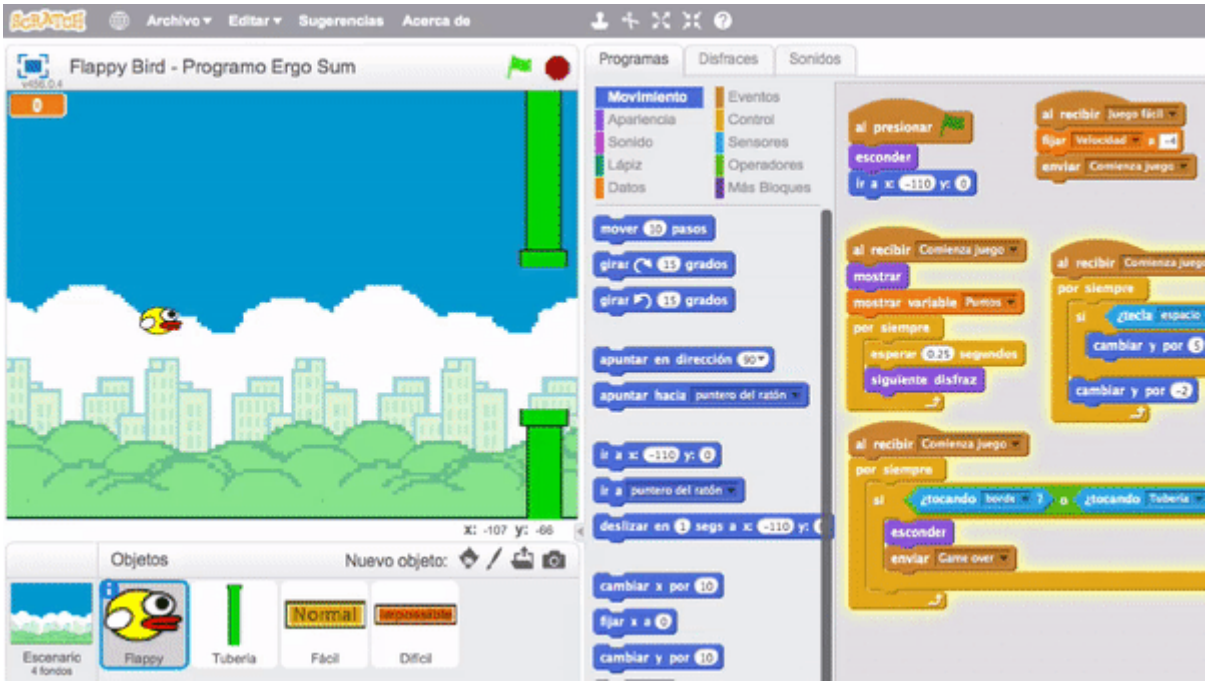
y finalizando

https://www.youtube.com/embed/zz0DTHV0XBE

Flappy movientos

Propuesta

La propuesta es de Programo Ergo Sum Flappy



Solución

Fondos

https://www.youtube.com/embed/LSMPzJ8x_GY

movientos

https://www.youtube.com/embed/cWvQ0d10wdE

puntuación

https://www.youtube.com/embed/PHadoJxg3Uo

Pong

Propuesta

https://www.youtube.com/embed/-ZJIcoFKG18

Solución

https://www.youtube.com/embed/-S5TDAg2LIw

https://www.youtube.com/embed/mfNMeEo0cHI

https://www.youtube.com/embed/ze-cvFf5DfE

https://www.youtube.com/embed/qYQzBsWAmhU

Rebotes - Arkanoid

Propuesta

Nuestra propuesta es de Programo Ergo Sum -arkanoid

Solución

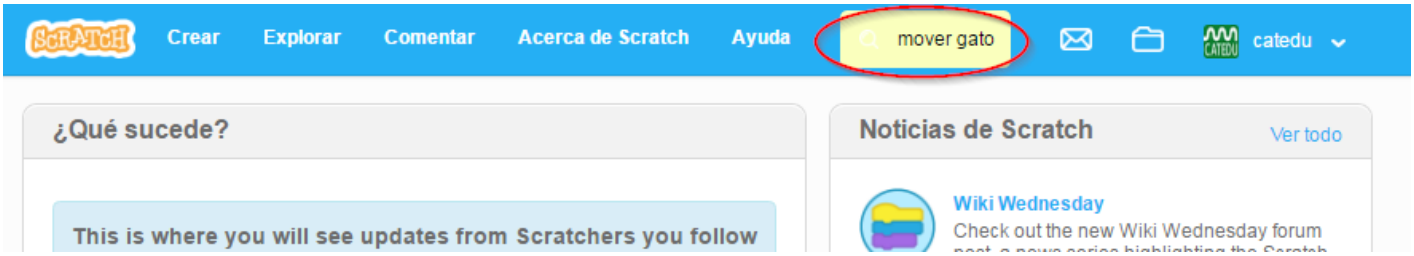
https://www.youtube.com/embed/dddeFS44f-E

https://www.youtube.com/embed/R1J6csAkbfs

https://www.youtube.com/embed/rN-eu3N29FU

Reinventar

Buscamos un programa que esté relacionado con el movimiento del sprit, por ejemplo:



y encontramos este https://scratch.mit.edu/projects/38947928/:

https://scratch.mit.edu/projects/embed/38947928/?autostart=false

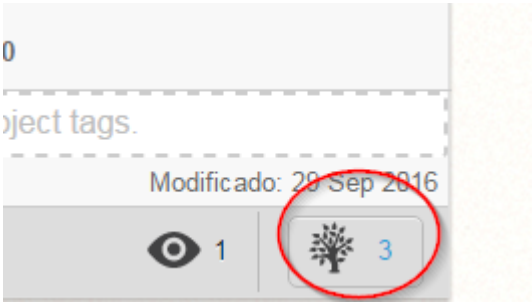
Nos gusta pero vemos que se podría mejorar:

Mejoramos el código, añadiendo una variable "frase_bienvenida" que recoja la respuesta y el texto que queremos que aparezca

Le damos a compartir y nos sale un mensaje de bienvenida y la opción de describir nuestra mejora:

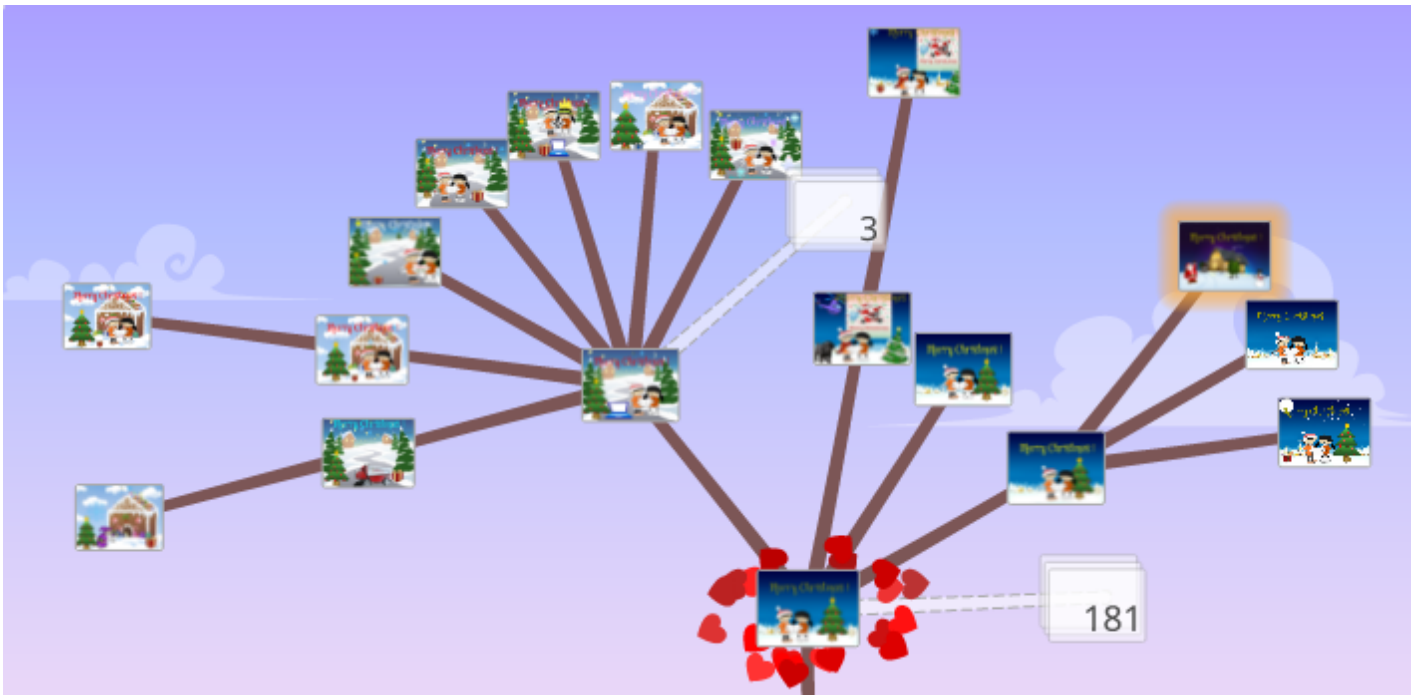
y ahora es otro proyecto: https://scratch.mit.edu/projects/123302249/

https://scratch.mit.edu/projects/embed/123302249/?autostart=false



Si pinchamos en el árbol sale esta página y aparece 3 reinvenciones (a fecha 29/9/16) el primero es el original, uno segundo (del mismo autor que lo mejoró) y esta última de Catedu

Hay proyectos donde las reinvenciones son muchas y salen muchas versiones de la misma, por ejemplo en este proyecto de tarjetas navideñas el árbol de reinvenciones



¿Por qué reinventar?

Recordemos que Scratch es código libre, por lo tanto es software libre de ser copiado, estudiado o modificado. Esto no debe de confundirse con OpenSource o código abierto, con licencia abierta para que técnicamente se mejore, o FreeWare, es decir gratis.

Aunque técnicamente no se esté obligado a mejorar el código como en OpenSource, sí que se obliga a acceder al código fuente para poder copiarlo y por lo tanto tenemos libertad de reinventar, podemos obtener una versión diferente que no necesariamente nos obligue a una mejora, somos libres de hacer con otros programas de otros usuarios lo que queramos mientras respetemos la licencia libre (acceso al código principalmente)

Se puede aprovechar la actividad de reinventar en el aula para introducir los conceptos de mejora de código comunitarios:

- Los 4 principios filosóficos del software libre. (ver)
- Los 10 principios técnicos del software abierto (ver)
- El software gratis no es libre o abierto y viceversa
- La mejora de la comunidad gracias al acceso de la licencias GNU o GPL es decir Licencia Publica
- La historia de GNU/Linux
- Desarrollo de programas libres frente a los privativos

	GNU	PROPIETARIO
SISTEMA OPERATIVO	Linux	Windows 8
NAVEGADOR	Mozilla Firefox	Google Chrome
SUITE OFIMÁTICA	LibreOffice Libre Office	Microsoft Office
REPRODUCTOR DE VIDEO	VLC media player	Reproductor Windows media player
REPRODUCTOR DE AUDIO	Songbird	iTunes
EDITOR DE IMÁGENES	Gimp	Photoshop
EDITOR DE VIDEOS	Avidemux	Adobe premiere pro

Conviertete en un Scratcher

Al cabo de un tiempo Scratch te preguntará si quieres ser un **Scratcher** ¿qué implica? pues básicamente que respetarás las normas básicas de convivencia un foro de este tipo, actualmente se fija en los siguientes aspectos:

- Limitaciones en tus **comentarios**: Tus críticas tienen que ser constructivas, no negativas y respetando las opiniones, trabajos y comentarios de los demás
- Limitaciones en **tiempo de respuesta** en los debates (dirigidos a evitar spam)
- Limitaciones en el uso de **enlaces externos**, imágenes, almacenamientos en la nube, etc... evidentemente sí que puedes usar enlaces a otros proyectos Scratch

El Scratcher, por supuesto tiene que estar registrado y aunque no es necesario, lo normal es que comparta proyectos, haga comentarios de otros trabajos, reinvente... es decir participar en la comunidad Scratch.

Aquí se puede trabajar los **valores de las comunidades virtuales**, la importancia de los comentarios, foros, debates ... sin perder la vista el **constructivismo**,**colaboración** ... no la simple palabrería. La cooperación, no sólo técnicamente, sino también se puede ser animar, felicitar ...

https://wiki.scratch.mit.edu/wiki/New_Scratcher_Status

Scratch para docentes

Abrir cuenta como profesor

Es distinto registrarnos como profesor que como alumno, incluso un usuario ya creado no puede convertirse en profesor, hay que empezar desde el principio.

Aquí tienes un tutorial en formato flash o en pdf si no lo ves bien:

Clase con Scratch

Pasado 24 horas, puedes averiguar si han aceptado tu cuenta de profesor y empezar a trabajar.

Para ver un poco las posibilidades, tienes este tutorial en FLASH o en PDF como siempre de elaboración propia:

También puedes verlo en este vídeo de Youtube de Scratch pero en inglés:

https://www.youtube.com/embed/7Hl9GxA1zwQ

Evaluar

Para evaluar un proyecto con Scratch podemos hacer una rúbrica más o menos elaborada como en este proyecto de música, pero implica un esfuerzo considerable. Proponemos:

Dr Scratch http://www.drscratch.org/

Es una forma objetiva, rápido aunque sólo evalúa el aspecto informático de programación, no los conceptos, la imaginación, los sprits y escenarios creados...

Podemos ingresar el proyecto mediante la URL (perfecto si utilizamos Scratch online) o por archivo (si utilizamos Scratch offline)

Evalúa del 0-3 estos items: Paralelismo, Pensamiento lógico, Control de flujo, Interactividad con el usuario, Representación de la información, Abstracción , Sincronización

En total del 0 al 21, incluso da una realimentación de mejoras

Si nos registramos, graba un historial de logros.

Un ejemplo:

Si evaluamos este proyecto https://scratch.mit.edu/projects/123355627/

https://scratch.mit.edu/projects/embed/123355627/?autostart=false

Si realizanos la evaluación en Dr Scratch: pues nos sale ... oh cielos !!! que malo que soy !!! por favor que esto no se publique !!!

Y encima da opción a descargarte un certificado en PDF ¡qué vergüenza !

Consejos

Para empezar Scratch en clase

- Fichas de Scratch descarga de la fuente
- Fichas de Scratch Aprende como un niño
- Los 12 retos cortos de menor a mayor profundidad descarga (doc - 337.5) fuente
- Para los más pequeños, con tableta y la app Scratchjr hacer estos retos
- Ideas para principantes en https://scratch.mit.edu/starter_projects/
- Paso a paso en Scratch online https://scratch.mit.edu/projects/editor/?tip_bar=getStarted

A la hora de proponer un videojuego en clase

Tal y como vistes en la presentación del módulo 1 :

Tiene que ser ejemplos que funcionen, adaptados a su capacidad. ¿Como se consigue esto?

- Entra en Scrach on line
- Mira ejemplos que funcionen
- Mira por dentro la programación si es adecuado a su capacidad
- A ellos propon la idea, de tal manera que sea difícil encontrar su solución exacta en Scratch

Fijar los objetivos ¿de que va?¿por qué es divertido?

Usar escenarios, diseñarlos, no entretenerse mucho en esto pues se escapa del pensamiento computacional.

Escribir la lógica en un papel, detectar inconsistencias, pedir ayuda.

Enlaces

Curso en EducaMadrid

- Prácticas básicas con Scratch 1.
- Prácticas básicas con Scratch 2.
- Prácticas básicas con Scratch 4.
- Prácticas básicas con Scratch 5.

Guias:

- En Eduteca
- Juan Carlos López 01/05/13
- Ceip Cella Teruel
- Gobierno de Canarias, con enlaces a diversas áreas Matemáticas, Lenguaje, Sociales
- Cefire de Elda

Retos

Aquí tienes un muro donde puedes encontrar todos aquellos ejemplos que veas interesantes para realizar en Scratch

https://padlet.com/embed/o2666arhnbxb

Hecho con Padlet

https://padlet.com/embed/phc0rpzhe1qj

Hecho con Padlet

Créditos

2017 por CATEDU (Daniel Pons Betrián y Javier Quintana Peiró).

Cualquier observación o detección de error en soporte.catedu.es

GOBIERNO DE ARAGON
Departamento de Educación, Cultura y Deporte

CATEDU
CENTRO ARAGONÉS de TECNOLOGÍAS para la EDUCACIÓN