UNIVERSIDAD DE VALLADOLID

DPTO. ÁLGEBRA, ANÁLISIS MATEMÁTICO, GEOMETRÍA Y TOPOLOGÍA.

SCRATCH para fomentar el razonamiento matemático

Trabajo Final del Máster Universitario de Profesor en Educación Secundaria Obligatoria y Bachillerato, Formación Profesional y Enseñanza de Idiomas. Especialidad de Matemáticas.

Alumno: Pablo Martín San José

Tutor: Philippe Thierry Giménez Martín

Valladolid, Junio de 2022

Índice general

Introducción

Las matemáticas son una disciplina centrada en la deducción y el razonamiento sobre temas abstractos. Una manera práctica de acercarnos a esto es la programación informática. En este trabajo proponemos distintas formas de hacerlo a un nivel de la ESO, utilizando para ello Scratch como herramienta didáctica. Veremos actividades centradas en varios cursos de la ESO, y aplicadas a la geometría, a la aritmética de los números enteros, y al más puro razonamiento presente en la lógica proposicional, tratando siempre de hacerlo de una manera que sea accesible y motivante para los alumnos.

No es nada nuevo que para muchos alumnos las matemáticas no son más que una asignatura aburrida, y que además no entienden, lo cual hace que les guste aún menos. Esto provoca una retroalimentación negativa, puesto que cuanto menor sea la motivación de un alumno en una materia, menor empeño pondrá en aprenderla y mayor será la frustración ante posibles resultados negativos, lo cual disminuye aún más la motivación por la materia.
Es posible que esto se deba al carácter "mecánico" que tienen las matemáticas que se ven en la ESO, centradas en la repetición de algoritmos de cálculo cuya única aplicación, en muchos casos, es el de poder realizar nuevos algoritmos de cálculo a partir de ellos.
Una solución posible a este problema podría ser acercar a los estudiantes a las matemáticas "de verdad", al razonamiento matemático, una habilidad que no solo es útil en matemáticas, sino en otros muchos aspectos tanto de la vida académica como profesional y personal de los alumnos.
Para conseguir este objetivo, tratamos de fomentar el razonamiento matemático en

la ESO mediante Scratch, una herramienta didáctica dirigida a niños de entre 8 y 16 años que pretende ser una manera sencilla y accesible de acercar a los alumnos a la programación, disponiendo de un entorno en el que pueden desarrollar tanto su creatividad como su capacidad analítica.

En el primer capítulo hablaremos sobre Scratch en general, tratando desde su historia y la filosofía de sus creadores, hasta el funcionamiento del programa en sí y las posibilidades que ofrece.

En el segundo capítulo tratamos más en concreto el uso de Scratch en el ámbito educativo, estudiando investigaciones que se han hecho al respecto. Primero hablaremos sobre Scratch como una herramienta didáctica general, en distintas áreas de la educación. Después nos centramos en nuestro área particular, hablando de la importancia de Scratch en la Didáctica de la Matemática. Por último, discutiremos la contribución de Scratch como herramienta didáctica a las competencias clave establecidas en el Real Decreto 217/2022, de 29 de marzo, por el que se establece la ordenación y las enseñanzas mínimas de la Educación Secundaria Obligatoria.

El tercer capítulo es dónde se sitúa el grueso del TFM. En él desarrollamos tres actividades distintas centradas en Scratch diseñadas para fomentar el razonamiento matemático.
La primera está planteada en cuarto de la ESO (en la asignatura de Matemáticas Orientadas a las Enseñanzas Académicas), y consiste en la resolución de una serie de acertijos lógicos, para los que los alumnos tendrán que inferir conclusiones a partir de unas hipótesis que reciben.
La segunda es una actividad para primero de la ESO en la que los alumnos deben dibujar ciertas figuras geométricas en Scratch, para estudiar algunas de sus propiedades.
En la tercera y última actividad, destinada a una clase de segundo de ESO, los estudiantes se enfrentan a un reto de programación: escribir un algoritmo (en Scratch, por supuesto) que permita comprobar si un número natural dado es primo o no.

Por último, en el cuarto capítulo reflexionamos sobre las conclusiones del trabajo, discutiendo si creemos que se trata de una buena herramienta didáctica para fomentar el razonamiento matemático en la ESO o no.

En la elaboración de este trabajo hemos juntado los distintos conocimientos aprendidos en las asignaturas cursadas a lo largo del año de Máster, especialmente en aquellas pertenecientes al módulo específico. Destacamos, entre estas, las asignaturas de "Didáctica de la Matemática", "Diseño Curricular en Matemáticas" y "Metodología y Evaluación en Matemáticas", que constituyen el núcleo a partir del cual hemos diseñado las actividades del tercer capítulo.

Capítulo 1

El programa Scratch

1.1. Historia

Scratch es un lenguaje de programación diseñado como una herramienta dirigida a niños de entre 8 y 16 años que permite crear una gran variedad de animaciones, juegos, historias interactivas y algoritmos de una manera extremadamente sencilla. El programa inicia su desarrollo en el año 2003 por el grupo "Lifelong Kindergarten"del Media Lab del MIT, con la idea de crear un recurso educativo con las características descritas, puesto que por aquel entonces no había ninguna herramienta que tuviera estas posibilidades y fuera accesible. La primera versión que fue lanzada al público vio la luz en 2007 y, aunque era un programa con muchas menos características que la versión actual, tuvo bastante éxito y poco a poco fue aumentando su popularidad, así como las características de la aplicación y la página web.
El 9 de mayo de 2013 Scratch recibió una importante actualización, y la versión 2.0 fue lanzada, incluyendo por primera vez un editor online, y una interfaz de usuario completamente nueva. Para entonces, Scratch ya tenía millones de usuarios, pero siguió creciendo hasta alcanzar los 30 millones de usuarios en 2018. La última gran actualización que ha recibido es la versión 3.0, lanzada el 2 de enero de 2019.

El lema de Scratch es "Imagina, Programa, Comparte", y nos da una clara idea sobre la filosofía de sus creadores: el objetivo es que los usuarios de Scratch compartan sus proyectos en la página web para que puedan ser utilizados y modificados ("rein-

ventados", en palabras de sus desarrolladores) por toda la comunidad. De la misma manera, el software Scratch es completamente gratuito y también es una aplicación de código libre. Es decir, el código está disponible en la red y todo el mundo tiene derecho de tomarlo y utilizarlo como quiera.

Scratch es un programa accesible (tiene un editor offline que funciona en Windows, macOS, ChromeOS y Android, y además se puede utilizar desde los navegadores más comunes, tanto desde un ordenador como una tablet o teléfono móvil) y orientado a que haya una interacción fuerte entre distintos miembros de la comunidad. Gracias a esta mentalidad, tanto en la página web de Scratch como en otros muchos lugares de internet podemos encontrar una gran cantidad de guías sobre Scratch, ideas de posibles proyectos, y muchos recursos educativos para padres y profesores. Por nombrar algunos ejemplos:

- En la página web de Scratch tenemos un apartado con decenas de tutoriales, que tratan desde el funcionamiento más básico del programa hasta actividades más complejas como crear un videojuego:
  `https://scratch.mit.edu/projects/editor/?tutorial=all`
  Tenemos un resumen de todas estás actividades en un pdf donde aparecen en forma de tarjetas, para mejor accesibilidad:
  `https://resources.scratch.mit.edu/www/cards/es/scratch-cards-all.pdf`

- También en la web de Scratch podemos encontrar una página con algunos proyectos creados por el equipo de Scratch, susceptibles de ser modificados, dando lugar a infinidad de posibilidades:
  `https://scratch.mit.edu/starter-projects`

- Por terminar con la página web, contamos con un pdf dirigido a educadores en el que encontramos una gran variedad de actividades adaptadas para ser llevadas a un aula, organizadas de manera que funcionen en una clase de 60 minutos:
  `https://resources.scratch.mit.edu/www/guides/en/EducatorGuidesAll.pdf`

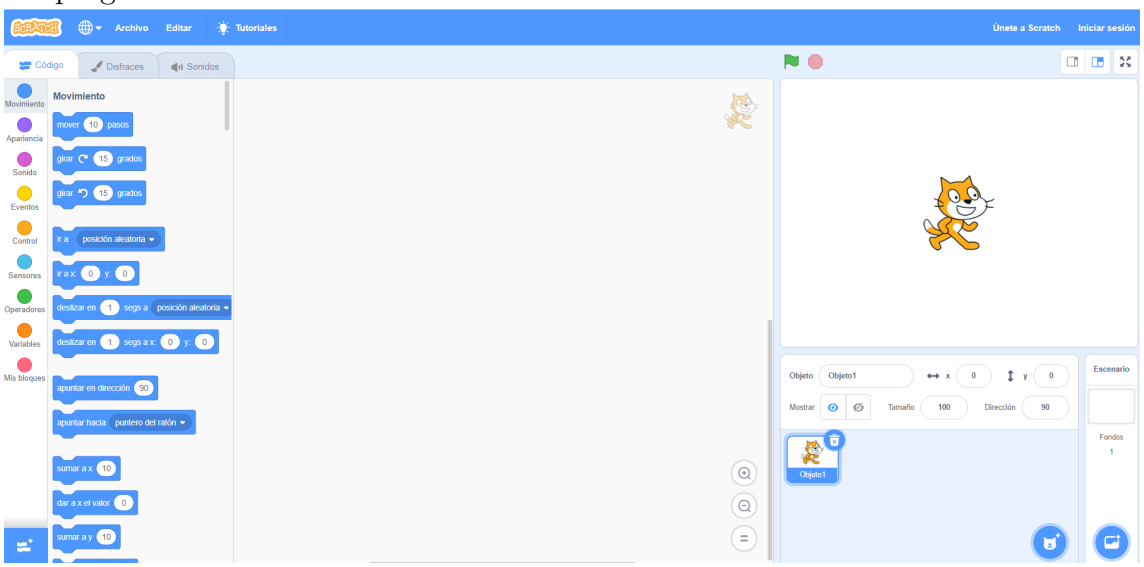
- El *Creative Computing Lab* de la *Harvard Graduate School of Education* ha creado el Creative Computing Curriculum, una colección de ideas, estrategias y actividades pensadas para una experiencia de computación creativa utilizando Scratch:
  `http://scratched.gse.harvard.edu/guide/`

Realmente la cantidad de recursos de Scratch que podemos encontrar en la web con una simple búsqueda en google (tanto en español como en inglés) parece infinita, así que no vamos a nombrarlos todos, pero merece la pena mencionar lo masiva que es la comunidad de usuarios de Scratch, porque esto contribuye a que la herramienta sea lo que es a día de hoy.

1.2. Funcionamiento

1.2.1. El editor

Ahora vamos a explicar las principales funcionalidades del editor online de Scratch (que es casi idéntico a la versión offline), para dar una idea sobre el aspecto y manejo del programa.



Esta es la pantalla inicial del editor de Scratch, en la que podemos observar varios elementos:

- La paleta de bloques:

Se encuentra en la parte izquierda de la pantalla (cuando seleccionamos la opción de "código, no la de "disfraces" o "sonidos"). Hay 9 categorías distintas de bloques (movimiento, apariencia, sonido, eventos, control, sensores, operadores, variables y mis bloques), de las que hablaremos más adelante. Los bloques son la manera que tenemos de programar en Scratch: un programa es una concatenación de bloques, que arrastramos a la zona de programación para utilizarlos.

- La zona de programación:
  Es el amplio espacio en blanco que encontramos a la derecha de la paleta de bloques. Aquí se almacenan todos los programas que correran en nuestro proyecto, formados por bloques. Cada uno de los objetos involucrados en nuestro proyecto tiene su propia zona de programación asociada, aunque los distintos objetos pueden "hablar" entre ellos.

- Los objetos:
  En la parte inferior derecha de la pantalla tenemos el editor de objetos, donde podemos ver los objetos presentes en el programa, cambiar sus parámetros (nombre, posición, si son visibles o no, su tamaño y dirección), o añadir nuevos objetos.
  Cada objeto tiene una imagen asociada, que determina cómo se ve en el escenario. Podemos cambiar está apariencia durante la propia ejecución del programa, cambiando el disfraz de este objeto. Podemos gestionar los distintos disfraces de un objeto desde la pestaña "disfraces"de la paleta de bloques.
  Los objetos también tienen asociados ciertos sonidos que pueden reproducir. Desde la pestaña "sonido" de la paleta de bloques podemos añadir más sonidos para un mismo objeto, ya sea importando los archivos de audio desde nuestro ordenador o grabando uno nuevo con nuestro micrófono, algo que nos permite hacer el editor sin mucho esfuerzo, así como editar (a un nivel básico) los sonidos que tengamos guardados.
  Asimismo, como ya hemos comentado, cada objeto tiene asociados sus propios bloques de código, que determinan lo que hace el objeto cuando ejecutamos

el proyecto. Podemos añadir o eliminar todos los objetos que queramos de nuestro proyecto, pero hay uno que siempre está presente: el escenario.

- El escenario:

  Cuando hablamos del escenario nos referimos tanto al objeto que tiene este nombre y que siempre forma parte de un proyecto, como a la parte superior derecha de la interfaz de usuario, que será donde se muestren los resultados de nuestro proyecto cuando lo ejecutemos.

  El escenario es un objeto un tanto particular, muestra algunas diferencias con los demás objetos, por ejemplo:

  - Es estacionario, no se puede mover de sitio, ni podemos considerar su distancia a otro objeto (cosa que si podemos hacer con otros objetos).
  - No puede hablar. Es decir, no podemos insertar bloques del tipo "decir..." o "pensar..." en la zona de programación del escenario.
  - Solo puede haber un escenario: no lo podemos clonar ni renombrar.
  - Siempre es visible, no podemos decidir cuando se muestra y cuando no.
  - En vez de tener disfraces, como los otros objetos, el escenario tiene distintos fondos entre los que puede cambiar.

  Cuando nosotros ejecutemos un proyecto, los objetos aparecerán en el escenario, donde también podremos ver (si así lo deseamos) los valores de algunas de las variables del programa.

- Miscelánea:

  En la parte inferior izquierda tenemos un botón para añadir extensiones. Las extensiones son nuevas categorias de bloques que podemos añadir a Scratch para disponer de más funcionalidades.

  También disponemos, encima del escenario, de una bandera verde y un octógono rojo, que sirven como controles para iniciar y detener un programa, respectivamente. En la parte superior, en la barra azul, disponemos de varios botones, de izquierda a derecha:

- Un enlace a la página principal de Scratch: `https://scratch.mit.edu/`
- El botón para cambiar el idioma del editor. El software incluye 70 idiomas distintos que podemos seleccionar.
- Desde el botón "Archivo" podemos guardar el proyecto en el que estamos trabajando en nuestro ordenador, y también podemos cargar un proyecto que tengamos almacenado, para seguir trabajando en él.
- Con el botón "Editar" podemos restaurar el último objeto que hayamos eliminado, y también podemos activar el modo Turbo, una característica que permite que las animaciones sean más rápidas, lo cuál puede ser útil en algunos proyectos.
- Tenemos un enlace a la librería de tutoriales.
- Por último, a la derecha del todo tenemos la opción de registrarnos en Scratch o iniciar sesión en caso de que ya lo estemos. Esto añadirá nuevas opciones a la barra azul, de las que hablaremos más adelante.

1.2.2. Los bloques

Puesto que los bloques son las piezas que conforman los algoritmos de Scratch, vamos a explicar las principales funcionalidades que tienen estos, pues eso nos permite conocer que podemos hacer con Scratch.
Podemos dividir los bloques atendiendo a distintos criterios. El primero es la forma que tiene el bloque, lo que determina cual es su función:

1. Los bloques que tienen forma de "sombrero" son los que inician un algoritmo. Solo se pueden insertar nuevos bloques debajo de estos, no encima. Tienen en común que inician el algoritmo que tienen por debajo cuando se cumpla una cierta condición. Por ejemplo, que el usuario haga click en la bandera verde, o en el objeto en el que se encuentra el bloque.

2. La mayoría de bloques tienen forma de rectángulo con una hendidura arriba y otra abajo. Estos bloques pueden llevar otros bloques encajados tanto encima

como debajo de ellos, y se encargan de realizar acciones, como reproducir un sonido, mover un objeto de posición, o cambiar el valor de una variable.

3. Los bloques con forma de hexágono se corresponden con condiciones de tipo booleano, que pueden ser verdaderas o falsas. Por ejemplo, que un número sea mayor que otro, que un elemento esté en una cierta lista, o que el usuario esté presionando cierta tecla.

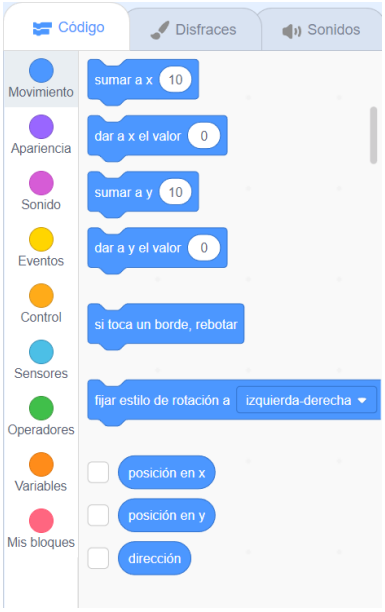
4. Muchos bloques tienen forma de "cápsula". Contienen un valor numérico o una cadena de caracteres.

5. Los bloques con forma de "C" permiten realizar bucles y condicionales en Scratch.

6. Por último, tenemos bloques que solo tienen una hendidura por encima, y que por lo tanto solo pueden ir debajo de otros bloques. Solo hay dos bloques de este tipo: uno se encarga de parar la ejecución de un algoritmo concreto o de todo el programa, y el otro elimina un clon de un objeto.

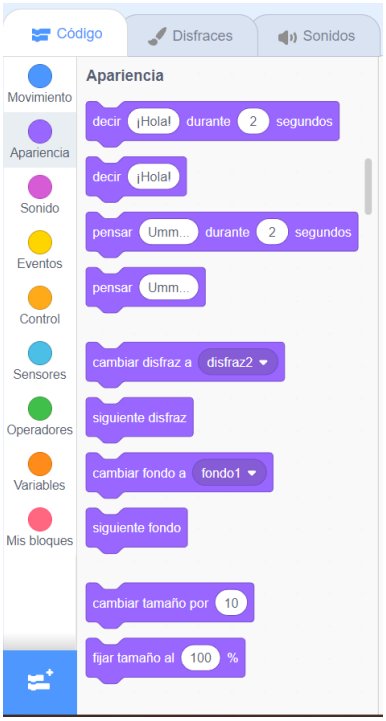
Para dar ejemplos de estos tipos de bloques vamos a mostrar una vista general de las 9 categorías de bloques que tenemos en Scratch:

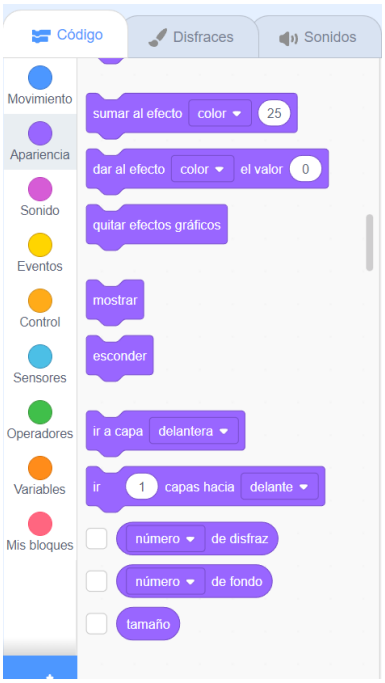
1. Movimiento:



La categoría de movimiento aparece asociada a todos los objetos salvo al escenario, y nos permite cambiar la posición y orientación de los mismos. Podemos observar que en los tres últimos bloques (las cápsulas relativas a posición en x, posición en y, y dirección) hay una casilla. Si la seleccionamos, el valor de esa variable se mostrará en el escenario en todo momento.

2. Apariencia:



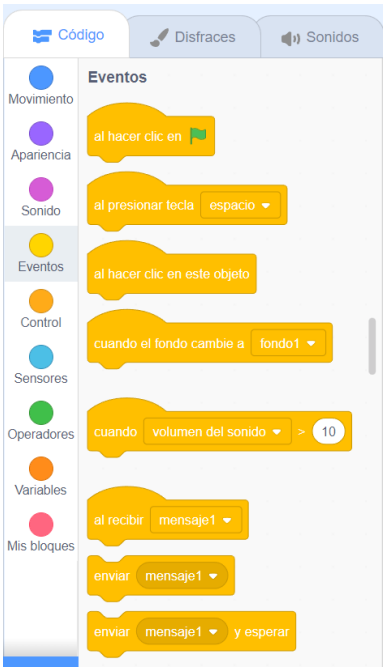


Los bloques de la sección de apariencia permiten que los objetos tengan cuadros de diálogo con mensajes de texto, que cambien su aspecto y tamaño, y también decidir si se muestran o no, y en cual de las capas. En caso de que dos objetos se solapen, el que este en la capa de más adelante será el que prevalezca.

3. Sonido:

En la categoría de "Sonido" encontramos bloques que reproducen (o detienen la reproducción de) sonidos asociados a un objeto, o que nos permiten poner efectos a los sonidos: mayor o menor volumen, y mayor o menor frecuencia.

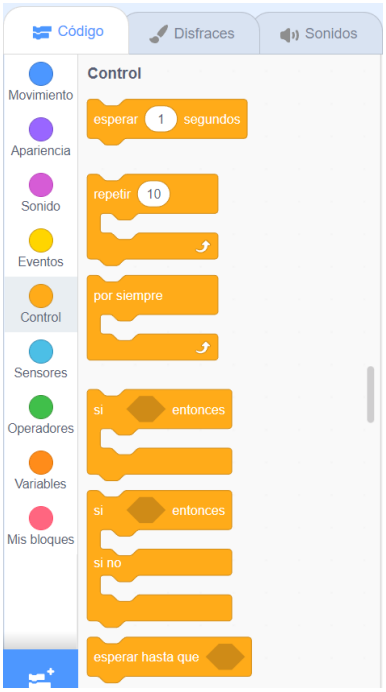
4. Eventos:

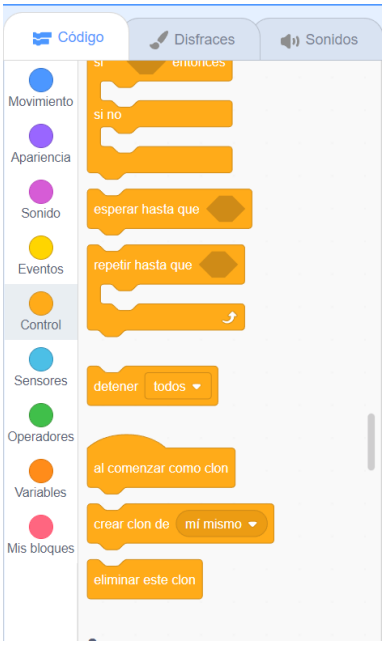


La categoría "Eventos" es de las más importantes. En ella se encuentran casi

todos los bloques que inician algoritmos, así como los bloques que permiten que los objetos se comuniquen entre sí: los referentes a los mensajes. Cuando un objeto envía un mensaje, cualquier otro objeto puede iniciar un algoritmo en consecuencia.

5. Control:

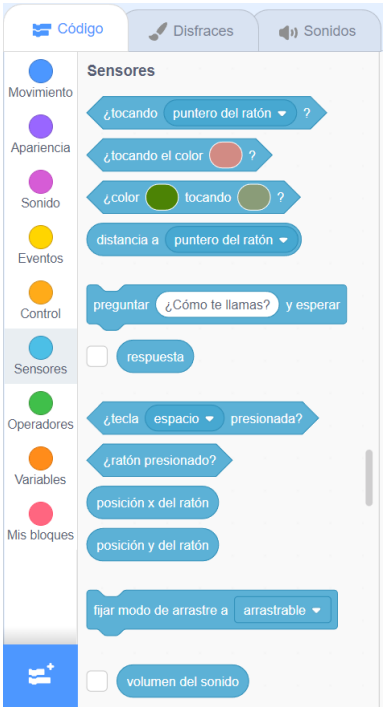


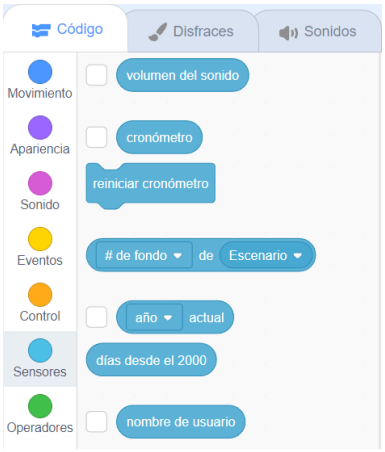


Desde el punto de vista de la programación, la categoría de control también es imprescindible, pues contiene bloques que permiten realizar bucles y condicionales. También contiene a los únicos dos bloques que pueden finalizar un algoritmo.

Sobre los bloques relativos a los clones, mencionaremos que lo que hacen es crear un objeto de aspecto idéntico al que contiene al algoritmo, y que mediante los bloques de esta categoría le podemos asociar otro comportamiento distinto al objeto original (o el mismo).

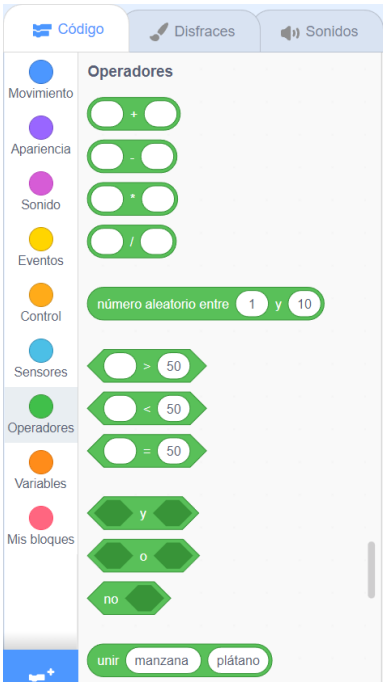
6. Sensores:





Esta categoría incluye una gran variedad de condiciones booleanas, así como maneras de registrar distintas entradas por parte del usuario.

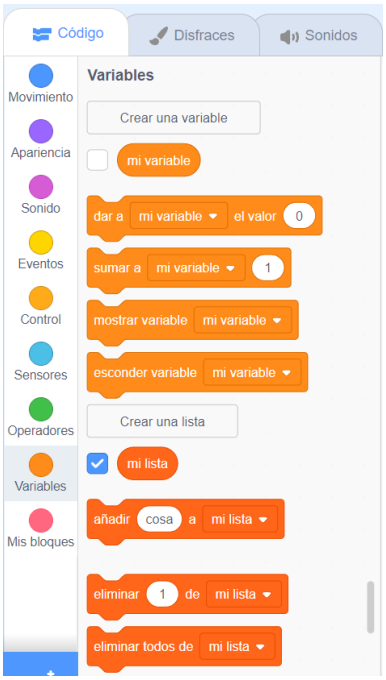
7. Operadores:

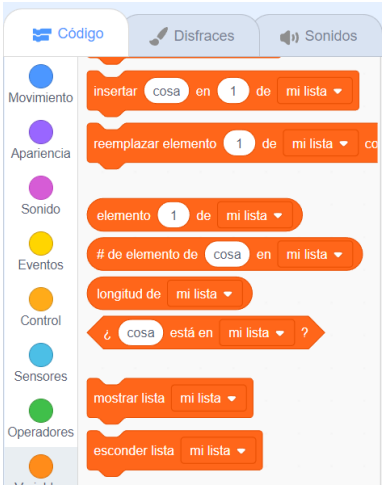


Los operadores son muy útiles desde el punto de vista matemático, y Scratch

incluye una gran variedad de funciones conocidas implementadas, lo cual es útil para poder programar algunos algoritmos de manera sencilla.

8. Variables:

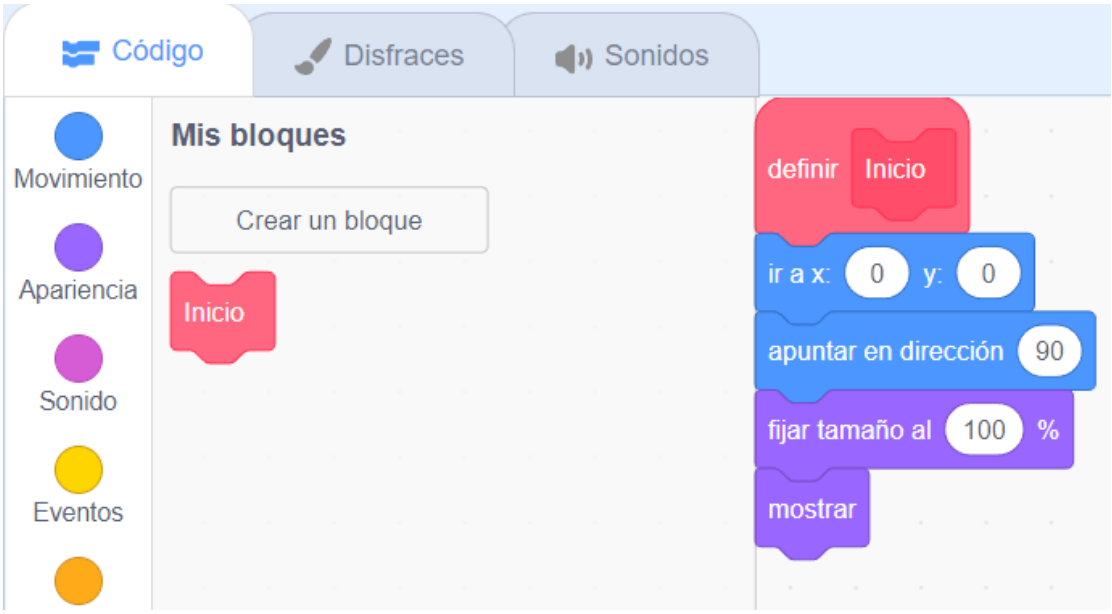




En esta categoría podemos definir nuevas variables y modificar sus valores. Hay dos tipos de variables: las que pueden tomar un valor numérico o de cadena de caracteres y las listas. Las listas son muy versátiles: incluyen una cantidad finita de elementos de cualquier tipo de manera ordenada, y podemos acceder a estos elementos mediante su posición y el nombre de la lista. Constituyen una implementación sencilla e intuitiva de los vectores que están presentes en algunos lenguajes de programación como C o Matlab.

9. Mis bloques:

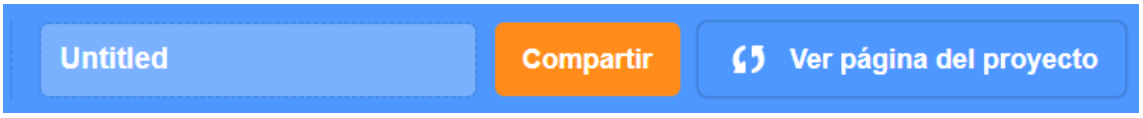
   Esta categoría esta vacía por defecto, pero nos permite crear bloques personalizados, que se correspondan a algoritmos creados por nosotros, y de esa manera podremos insertar trozos de código que se repitan de una manera limpia, así como mantener los programas más compactos. Este es un ejemplo de un bloque personalizado, que sirve para inicializar la posición, orientación y tamaño de un objeto de manera cómoda:



Después de definir este bloque personalizado, podemos insertar el bloque "Inicio"donde queramos para realizar esa secuencia de acciones, sin tener que insertar los 4 bloques correspondientes cada vez.

1.2.3. Compartir nuestros proyectos

Desde la página principal de Scratch tenemos acceso a la pestaña "explorar", donde podemos encontrar todos los proyectos compartidos por la comunidad, y filtrarlos según el género, la popularidad que tengan, o la fecha en que se hayan publicado.
Si estamos registrados en Scratch, nosotros también podremos compartir nuestros proyectos, como podemos ver aquí:



Podemos cambiar el nombre de nuestro proyecto, compartirlo, o visitar la página web asociada a nuestro proyecto, donde todo el mundo puede consultar y "reinventar" nuestro código.
Nosotros compartiremos todos los proyectos asociados a este TFM, para que así cualquier docente que desee aprovecharlos para sus clases pueda hacerlo de manera sencilla.

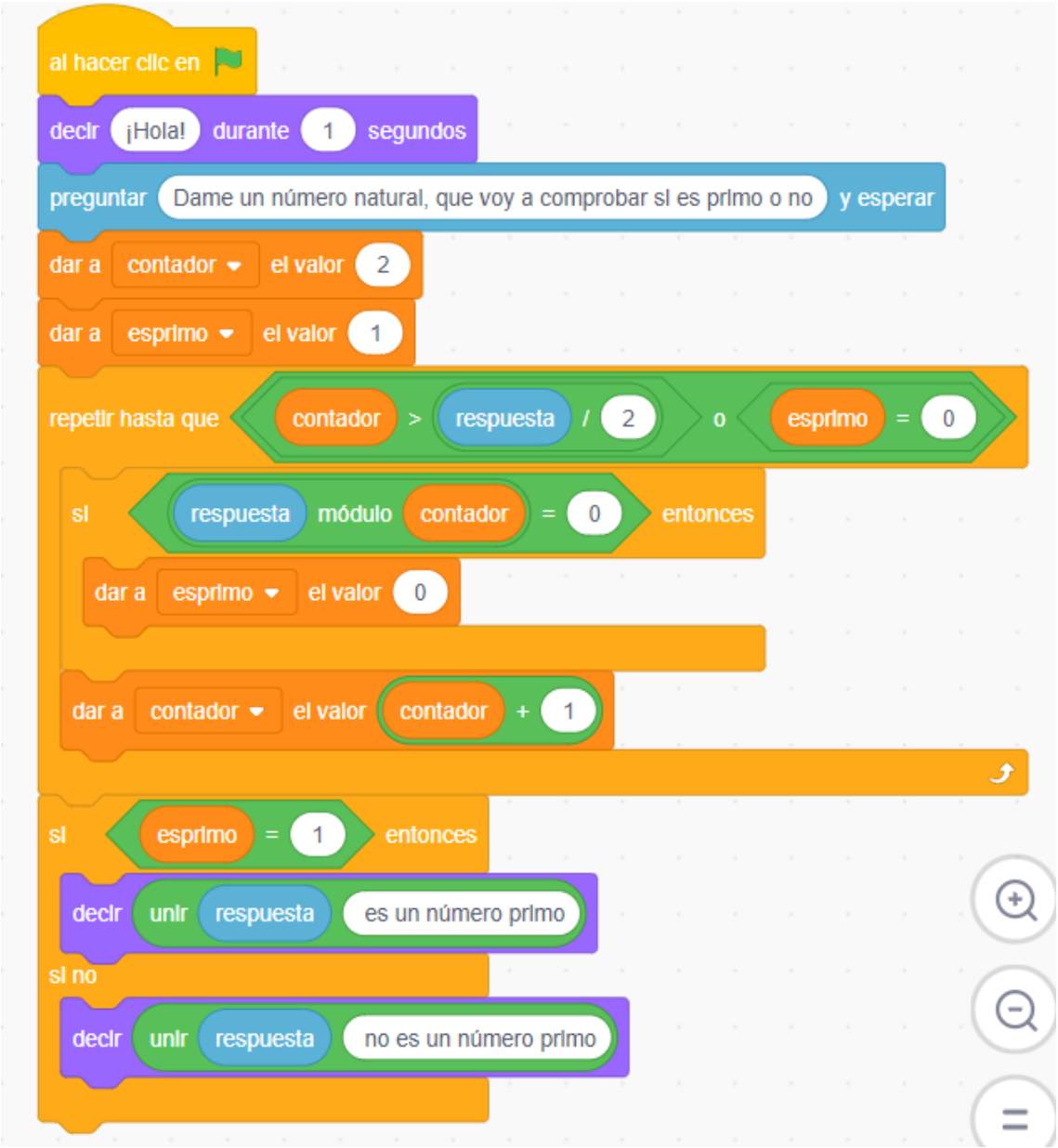
1.3. Posibilidades

No debemos dejar que la apariencia sencilla y funcionamiento por bloques de Scratch nos haga pensar que solo es una herramienta que ayuda a adolescentes a crear algunas animaciones, o historias interactivas y videojuegos. Scratch es un lenguaje de programación de verdad, con la mayoría de (si no todas) las características de los lenguajes de programación que se usan de manera profesional: bucles, condicionales, scripts, funciones, listas...

Podríamos decir que la única diferencia real entre Scratch y otros lenguajes de programación es la manera de escribir código. O, mejor dicho, el hecho de que en Scratch no escribimos el código, introducimos las instrucciones mediante los bloques. Esto no es algo negativo, desde luego, sino todo lo contrario, pues hace que mientras estamos teniendo nuestro primer contacto con la programación podamos centrarnos solo en los algoritmos, dejando la sintaxis de lado. Esto es muy similar a la idea detrás de los diagramas de flujo, que se utilizan para tener una idea de lo que debe hacer un programa, antes de escribirlo en un lenguaje real.

De hecho, a modo de curiosidad, mencionamos que Scratch es un sistema Turing completo. Es decir, es capaz de simular a una máquina de Turing (un modelo matemático de un ordenador). Esto quiere decir que, teóricamente, podemos programar cualquier algoritmo que pueda ejecutar un ordenador en Scratch.

Aquí tenemos, por ejemplo, una implementación sencilla del algoritmo que comprueba si un número es primo o no:



`https://scratch.mit.edu/projects/699474244`

Más adelante utilizaremos este programa para una de nuestras actividades.

Capítulo 2

Scratch en el ámbito educativo

2.1. Scratch como herramienta didáctica

Entendemos como herramienta didáctica un elemento, recurso o material que puede utilizar un docente para enseñar, favoreciendo el aprendizaje de los alumnos. Como veremos, hay diversos estudios que consideran que Scratch tiene muchos beneficios en la enseñanza, y es por eso que consideraremos que Scratch es una herramienta didáctica.

Podemos encontrar ejemplos de aplicaciones de Scratch en la educación desde practicamente sus orígenes. Por ejemplo, Calder y Taylor (2010) tienen una investigación en la que un grupo de 26 alumnos de sexto de primaria deben diseñar un juego de matemáticas que ayude a sus compañeros de primero de primaria a entender los números. Encontraron que Scratch proporciona un ambiente de programación motivante para explorar algunas ideas matemáticas. López-Escribano y Sánchez Montoya (2015) proporcionan varios ejemplos de uso de Scratch en alumnos con necesidades educativas especiales, e insisten sobre las ventajas de su utilización, puesto que es una experiencia lúdica y divertida, beneficiosa para todos los alumnos.

Sobre las habilidades que se desarrollan en los alumnos mediante el uso de Scratch podemos destacar la creatividad, lo cual es de esperar, puesto que crear un proyecto de Scratch involucra una amplia variedad de decisiones de diseño, que pueden

ser afrontadas de maneras muy distintas. Prueba de ello es el estudio realizado por Kobsiripat (2015), donde se concluye que el aprendizaje de programación mediante Scratch proporciona mayor nivel de flexibilidad y capacidad de tener ideas ingeniosas. Por otra parte hay múltiples estudios que afirman que Scratch es muy beneficioso para fomentar el pensamiento de tipo computacional o lógico-matemático. Así lo corroboran Calao, Moreno-León, Correa y Robles (2015), que encontraron "una ganancia estadísticamente significativa en la comprensión del conocimiento matemático en los alumnos que habían recibido formación en Scratch". Zhang y Nouri (2019) realizan una revisión exhaustiva de artículos existentes sobre el desarrollo de ciertas habilidades de pensamiento computacional (bucles, condicionales, paralelismo, variables...), donde concluyen que Scratch ayuda a asentar las bases de muchos de estos conceptos.

Es posible que lo que haga de Scratch una herramienta tan buena para desarrollar las habilidades anteriores es que contribuye a la motivación de los alumnos, lo cual les hace más susceptibles de aprender, y de querer trabajar por su propia cuenta sobre las actividades de clase. El estudio realizado por Nikou y Economides (2014) sugiere que Scratch aumenta la motivación para involucrarse en la programación y la autoeficacia (es decir, la confianza en la capacidad de uno mismo para lograr unos objetivos) de los alumnos, lo cual inevitablemente lleva a un mayor nivel de compromiso con las tareas y un mejor desempeño en las mismas.

Además de todas estas ventajas intrínsecas de Scratch, merece la pena insistir de nuevo en su gran accesibilidad. En el mundo actual, donde casi todo adolescente tiene un teléfono móvil inteligente a su disposición, ni siquiera será necesario disponer de un aula de informática para una sesión de clase en la que se implemente Scratch. Con un proyector digital para el profesor y un teléfono, tablet u ordenador para cada alumno (materiales con los que cuentan la mayoría de institutos) es suficiente en una buena parte de los casos. Asimismo, Scratch es accesible en el sentido de que se adapta bien a las necesidades de alumnos con discapacidades motrices o auditivas (no así a los que presenten una alta deficiencia visual), por lo que hace un gran trabajo de atención a la diversidad.

Queda claro que Scratch es una buena herramienta didáctica para trabajar en matemáticas e informática. Si bien generalmente se le asocia con estas materias, puede ser útil en una gran cantidad de disciplinas, es un recurso realmente versátil:

- Plástica: en Scratch podemos crear nuestros propios escenarios y objetos, dándoles el aspecto que más nos guste. Se puede utilizar esto para crear animaciones personalizadas y dar rienda suelta a la creatividad.
- Dibujo técnico: con la extensión "Lápiz" podemos dibujar la trayectoria de un objeto. Combinando esto con los bloques de movimiento podemos dibujar muchas de las figuras planas que se estudian en esta asignatura, comprendiéndolas desde otra perspectiva.
- Física: como muestran Lopez y Hernandez (2015), se puede utilizar Scratch para representar algunos modelos físicos, algo muy importante en esta materia.
- Lengua y enseñanza de idiomas: a través de Scratch se pueden crear historias interactivas, desarrollando así la capacidad de escribir, la ortografía, el desarrollo de personajes, recursos literarios...

Desde luego las posibilidades son infinitas. A veces el motivo por el cual no se implementan estas metodologías es por falta de formación del profesorado en este tipo de recursos. Este TFM contiene algunas ideas sobre como implementar Scratch en una clase de matemáticas de secundaria, por lo que en parte buscamos ayudar en este problema.

2.2. Scratch en la Didáctica de la Matemática

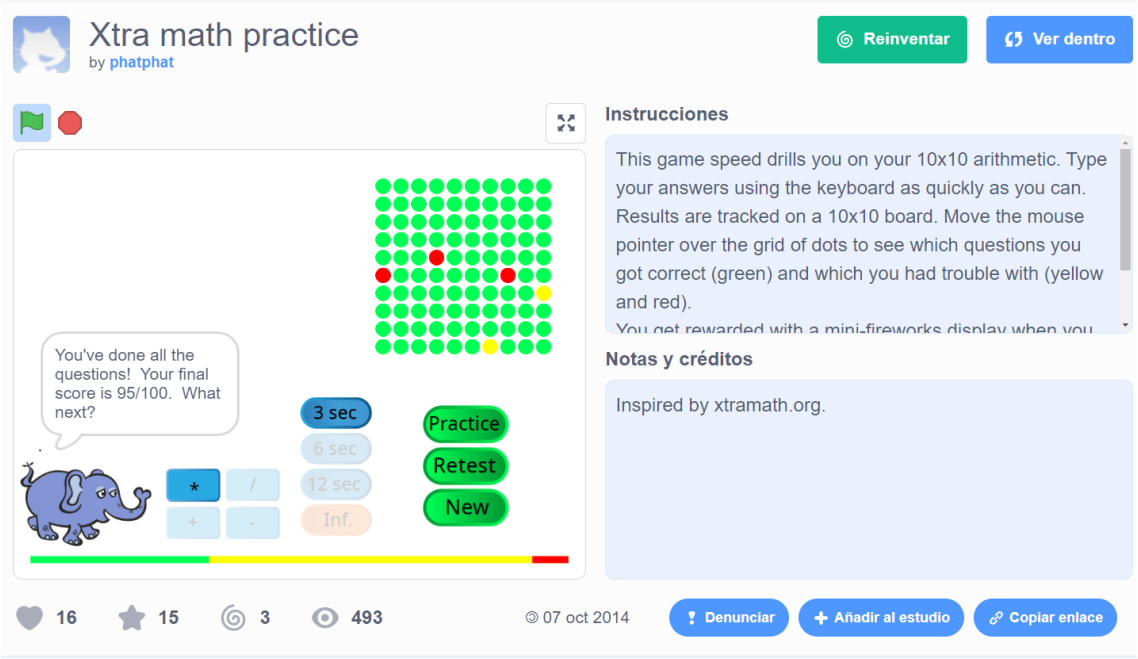
Ahora que hemos hablado de Scratch como una herramienta didáctica en general, vamos a centrar nuestra atención sobre el papel que juega como herramienta didáctica en el área de las matemáticas.

Una primera característica que podemos destacar de Scratch como herramienta didáctica es que favorece un aprendizaje activo, pues fomenta que los alumnos prueben por ellos mismos lo que hace cada bloque, y el efecto que tiene en los objetos y

el escenario. Esto es fundamental en la Didáctica de la Matemática, puesto que la manera en la que habitualmente uno se familiariza con los conceptos matemáticos es esencialmente esa, experimentando y probando varias ideas.

Otro aspecto a resaltar sobre Scratch es que es una herramienta basada en las Tecnologías de la Información y la Comunicación. Esto no es una garantía de éxito en si misma, pero sí que es cierto que este tipo de tecnologías a veces son más cercanas a los alumnos que otros recursos más "analógicos", y pueden aumentar la participación de estos. Además en las matemáticas actuales es muy importante el aspecto computacional (algunos teoremas, como el teorema de los 4 colores o la conjetura de Keppler, tienen pruebas realizadas mediante el apoyo de un ordenador). Por lo tanto, cuanto antes se acerquen los alumnos a la programación, mejor competencia matemática desarrollaran en un futuro, no solo por el hecho de tener una cierta soltura en el desarrollo de algoritmos, sino también porque el pensamiento computacional favorece la capacidad de razonamiento matemático.

Por último, mencionar que muchas actividades diseñadas en Scratch son una forma de aprendizaje basado en juegos, y de gamificación. Esto puede ayudar con el aspecto más tedioso y rutinario de las matemáticas: la necesidad de repetir ciertos procedimientos con el fin de ganar familiaridad con un tipo de cálculo que se va a utilizar frecuentemente. Un ejemplo de esto es el siguiente proyecto, que sirve para ganar soltura con las operaciones aritméticas básicas:



`https://scratch.mit.edu/projects/28560930/`

El juego funciona de la siguiente forma: elegimos una de las 4 operaciones básicas (suma, resta, multiplicación o división), y un intervalo máximo de tiempo que queremos para responder a cada pregunta. Después el juego nos somete a una batería de 100 operaciones de este tipo (con descansos cada 25 operaciones), que debemos completar en el menor tiempo posible. Al final de la partida podremos ver en un tablero de 10x10 nuestros resultados. Las preguntas que hayamos respondido correctamente aparecerán en verde, las que hayamos contestado bien pero fuera de tiempo aparecerán en amarillo, y las que hayamos respondido incorrectamente o no hayamos respondido aparecerán en rojo. De esta manera, al final de cada partida podemos saber donde están nuestros puntos débiles, y así mejorar para intentar obtener cada vez una mejor puntuación.

Podemos encontrar muchos ejemplos de proyectos como el anterior, que nos muestran que Scratch es una gran herramienta para desarrollar contenidos matemáticos que van mas allá de los algoritmos y la programación. Algunos que hemos encontrado y creemos que merece la pena mencionar son:

- Aritmética: Aparte del programa que mencionamos anteriormente, hay una gran variedad de proyectos para practicar las operaciones aritméticas de una

manera entretenida. En `https://scratch.mit.edu/projects/32386742/` podemos practicar el orden de operaciones mediante una historia peculiar, pero que puede entretener a algunos alumnos de primero de la ESO que tengan dificultades con este tema. Aunque `https://scratch.mit.edu/projects/205911151/` puede estar más enfocado a que niños de primaria practiquen con operaciones aritméticas, hemos decidido incluirlo puesto que incentiva a conseguir la mejor puntuación, lo cual puede ser una buena motivación para aprender. Por último, en `https://scratch.mit.edu/projects/1172016/` encontramos una aplicación en la que podemos realizar un test personalizado en el que decidimos que temas queremos tratar (entre sumas, restas, multiplicaciones, divisiones y álgebra), la dificultad y cantidad de preguntas, y al final podemos saber que preguntas hemos acertado.

- Geometría: Si buscamos la palabra clave "Geometría" encontraremos muchos proyectos interactivos que ayudan a memorizar y trabajar algunas de las fórmulas para el área de figuras planas y algunos otros tienen animaciones de figuras geométricas muy interesantes. Destacaremos al usuario jotace86 (`https://scratch.mit.edu/users/jotace86/`), que ha realizado un proyecto para introducir el concepto de sistema de referencia en un plano, y otro proyecto en el que debemos resolver 5 cuestiones de geometría, en el contexto de una divertida historia.

- Razonamiento matemático: La comunidad de Scratch ha programado una gran variedad de juegos con los que desarrollar el razonamiento matemático a la vez que nos divertimos. El proyecto `https://scratch.mit.edu/projects/527174167` es un bot del juego "4 en raya", que no es fácil de vencer si no pensamos nuestros movimientos y sus consecuencias. En el juego `https://scratch.mit.edu/projects/336228232/` debemos pensar en el camino que tomaran unos pájaros que coloquemos en un tablero con casillas que tienen distintos efectos. El objetivo es que cada pájaro recoja una llave situada en el tablero y llegue hasta una jaula en la que hay otro pájaro atrapado.

Con todo esto, podemos decir con cierta confianza que Scratch es una herramienta

didáctica que puede divertir a los alumnos y motivarles para aprender matemáticas en las etapas educativas de primaria y la ESO. Por otro lado, puede servir a los docentes para trabajar algunas materias de una nueva manera, permitiendo salir de la rutina e introducir cierta variedad en las clases.

2.3. Contribución a las competencias clave

El objetivo principal de esta sección es el de especificar en que manera una enseñanza de las matemáticas apoyada en Scratch puede desarrollar las ocho competencias clave establecidas en el Real Decreto 217/2022, de 29 de marzo, por el que se establece la ordenación y las enseñanzas mínimas de la Educación Secundaria Obligatoria.

1. Competencia en comunicación lingüística:
   Los alumnos están fortaleciendo su competencia en comunicación lingüística en el mismo momento en el que tienen que expresar de manera oral o escrita el razonamiento que hay detrás de la resolución de un problema, o la idea detrás de un algoritmo que hayan programado. Es esencial desarrollar esta habilidad en matemáticas, puesto que una idea correcta que sea mal expresada no será comprendida por los demás. Encomendando a los alumnos la tarea de elaborar un programa en Scratch que explique e ilustre un concepto matemático es una forma directa de trabajar esta competencia, por ejemplo.

2. Competencia plurilingüe: Scratch está traducido a 70 idiomas distintos. Por lo tanto se adapta muy bien a enseñar las matemáticas en otra lengua: si estamos en un instituto bilingüe, en el que la enseñanza de las matemáticas se realiza en otro idioma, solo tenemos que darle a un botón para que todos los bloques cambien a ese idioma. Por otro lado, si tomamos proyectos hechos por la comunidad, hay una gran cantidad de ellos que están hechos en inglés, es el idioma más común entre los programas de la web. Estar en contacto con otras lenguas, y posiblemente desarrollar algún proyecto en ellas refuerza la competencia plurilingüe, y vemos que esto es muy fácil de hacer con Scratch.

3. Competencia matemática y competencia en ciencia, tecnología e ingeniería:

Es evidente que las actividades que se presentarán en este TFM están pensadas, principalmente, para fortalecer la competencia matemática, como indica el título del mismo. Se intentará que los alumnos desarrollen su capacidad analítica: para resolver un problema de matemáticas hay que tener una visión global de este, y realizar procesos inductivos y deductivos para conocer nueva información a partir de las hipótesis hasta llegar a la solución. En todo este proceso es crucial la competencia matemática.

4. Competencia digital: Al ser Scratch una herramienta didáctica completamente basada en las Tecnologías de la Información y la Comunicación, se trabaja constantemente la competencia digital. Los alumnos están en contacto con un software informático en un entorno web, y deben familiarizarse con sus controles básicos: crear un nuevo proyecto, guardarlo en su cuenta personal, compartirlo con la comunidad, buscar y reinventar un proyecto que ya esté hecho, etc.

5. Competencia personal, social y de aprender a aprender: Para conseguir potenciar esta competencia incentivaremos que los alumnos tengan un cierto nivel de trabajo personal a la hora de desarrollar los proyectos de Scratch por su cuenta. Con esto crearemos un hábito y disciplina de trabajo, los alumnos serán capaces de reflexionar sobre su forma de trabajo, y adaptarla de la mejor manera posible a sus características personales para obtener buenos resultados.

6. Competencia ciudadana: Fomentamos la competencia ciudadana en una clase de Secundaria en todo momento, realmente, porque los alumnos conviven entre sí, y con los profesores, por lo que frecuentemente se deben resolver conflictos que surjan, y trabajar por los intereses de todo el mundo. En nuestras actividades de Scratch, en concreto, se propondrán metodologías colaborativas, en las que los alumnos deban cooperar y trabajar en grupo para conseguir sus objetivos, dando lugar a situaciones en las que se trabaje la competencia ciudadana.

7. Competencia emprendedora: Por un lado es fundamental que los alumnos sean

capaces de tomar la iniciativa a la hora de resolver un problema de matemáticas: tomar los recursos que tengan a mano y hacer el mejor uso posible de los mismos para dar con la solución.

Por otra parte, Scratch es un entorno muy orientado a colaborar con otros miembros de la comunidad, compartiendo proyectos, trabajando en conjunto con más usuarios, etc. Motivando a los alumnos a tener un cierto gusto por la comunidad de Scratch, estamos también incitándolos a que participen en estos proyectos comunitarios: por ejemplo, un alumno realmente apasionado sobre Scratch podría crear su propio estudio y publicar regularmente sus proyectos personales.

8. Competencia en conciencia y expresión culturales: Para justificar la consecución de esta competencia aludimos al hecho de que Scratch es un medio donde los alumnos pueden expresar su creatividad, por ejemplo mediante la creación de historias, y conociendo las creaciones de otros usuarios. Es de esta manera, mediante el intercambio de elementos culturales, que los alumnos pueden conocer diversos contextos e ideas del resto de sus compañeros de clase, desarrollando así "conciencia y expresión culturales".

Capítulo 3

Actividades basadas en Scratch para la ESO

Nos hemos adaptado al currículo de matemáticas de la ESO que se presenta en la ORDEN EDU/362/2015, de 4 de mayo, por la que se establece el currículo y se regula la implantación, evaluación y desarrollo de la educación secundaria obligatoria en la Comunidad de Castilla y León (2015), que aparece en el BOCyL núm. 86, de 08/05/2015.

3.1. Actividad 1: Los cofres de Porcia

Esta actividad es una adaptación del quinto capítulo del libro de Smullyan (2008), un libro con más de 200 acertijos lógicos y problemas adecuados para todas las edades, que van desde simples razonamientos de una linea hasta las más profundas paradojas de la teoría de conjuntos y la lógica matemática, explorando temas como la paradoja de Russell o el teorema de incompletitud de Gödel.

Este capítulo nos cuenta la historia de Porcia, un personaje de la obra "El mercader de Venecia", de Shakeaspeare. En ella, los pretendientes de Porcia debían elegir entre tres cofres, uno de oro, uno de plata, y uno de plomo, en uno de los cuales se encontraba el retrato de Porcia. Si el pretendiente elegía el cofre correcto, podía casarse con Porcia. Raymond Smullyan le da un giro a este concepto, y propone algunos acertijos partiendo de la misma idea de los cofres, pero añadiendo el hecho

de que cada cofre tiene una inscripción con información sobre dónde se encuentra el retrato. Con estas inscripciones, junto con cierta información sobre la veracidad de las mismas (por ejemplo, una pista podría ser "Solo hay una inscripción verdadera") se puede deducir donde se encuentra el retrato, y de esta manera Porcia puede elegir con qué pretendiente se va a casar en función de su inteligencia.
En nuestra actividad hemos incluido solamente los siete primeros acertijos de este capítulo. Los dos últimos no se adaptan tan bien a lo que tratamos de enseñar, pues en ellos, Porcia deliberadamente no da información a los pretendientes sobre las inscripciones, lo que hace que siempre elijan el cofre incorrecto.
Esta actividad está pensada como una primera introducción a la lógica proposicional, uno de los sistemas formales más sencillos que se pueden estudiar, pero que cuenta con una gran utilidad. Consideramos que, a pesar de que este tema no se trata casi nunca en la asignatura de matemáticas, puede aportar grandes progresos al razonamiento lógico de unos alumnos de cuarto curso de la ESO, situados en la asignatura de Matemáticas Orientadas a las Enseñanzas Académicas.

3.1.1. Contenidos

Los contenidos del currículo que desarrollamos con esta actividad son, principalmente, los correspondientes al Bloque 1, de contenidos comunes. Por citar algunos:

- *Planificación del proceso de resolución de problemas: análisis de la situación, selección y relación entre los datos, selección y aplicación de las estrategias de resolución adecuadas, análisis de las soluciones y, en su caso, ampliación del problema inicial.*

  En los siete acertijos a resolver, los alumnos deben ser capaces de analizar correctamente las premisas de las que disponen, y reflexionar sobre sus consecuencias lógicas, para así poder dar con la solución correcta.

- *Reflexión sobre los resultados: revisión de las operaciones utilizadas, asignación de unidades a los resultados, comprobación e interpretación de las soluciones en el contexto de la situación, búsqueda de otras formas de resolución, etc.*

Como veremos más adelante, se va a premiar a los alumnos que no hagan respuestas incorrectas a los acertijos, incentivando así la revisión de la respuesta y del razonamiento detrás de ella. De esta manera, los alumnos están obligados a asegurarse de que su solución sea correcta antes de introducirla en la aplicación.

- *Confianza en las propias capacidades para desarrollar actitudes adecuadas y afrontar las dificultades propias del trabajo científico.*
  Por un lado, el disponer de una mayor capacidad de razonamiento lógico aporta confianza en las capacidades propias del alumnado a la hora de resolver nuevos problemas que se les presenten. Por otro lado, es raro que los estudiantes no se queden atascados en algún acertijo. Esto es importante porque la capacidad de lidiar con la frustración y seguir intentándolo sin rendirse es una habilidad muy importante en el trabajo científico.

Aparte de los contenidos que aparecen explícitamente en el currículo presente en el BOCyL, vamos a desarrollar otros que no están presentes, referidos a la lógica proposicional como tal, que reunimos aquí:

Definición. Una proposición lógica es una afirmación referida a objetos conocidos, y para la cual podemos determinar sin lugar a dudas que es lo que significa que sea cierta o falsa.

Ejemplo. Ahora incluimos algunos ejemplos en los que podremos distinguir entre qué tipos de oraciones se corresponden con proposiciones y cuales no:

- La oración "Uno más uno es igual a dos" es una proposición lógica, puesto que sabemos que lo que afirma es exactamente la igualdad entre dos números naturales (el 1+1 y el 2), unos objetos que nos son conocidos, y nosotros sabemos sin lugar a dudas lo que significa que dos números naturales sean iguales o no. Un matemático habitualmente escribiría esta proposición como "$1 + 1 = 2$". En este caso estamos ante una proposición verdadera.

- La afirmación "$1 + 1 = 7$", a pesar de ser falsa, también es una proposición, por el mismo motivo que lo era la anterior.

- La frase "El número 14" no es una proposición, no porque los objetos que involucra no nos sean conocidos, sino porque no es una *afirmación*, en ella no afirmamos nada sobre lo que podamos juzgar su veracidad o falsedad.

- La afirmación "La Didáctica de la Matemática es aburrida" no es una proposición, porque aunque afirma algo que podríamos juzgar si es cierto o falso, no podemos hacerlo de manera inequívoca, puesto que el término "aburrido" es subjetivo: para unas personas significa una cosa, y para otras personas otra.

- La oración "Esta oración es falsa" no es una proposición. Esto es debido a que incluye un término (falsa) para el que no conocemos una definición universal, y que nos impide comprender cual sería el significado de que esta frase fuera cierta o falsa. Si fuera cierta sería falsa, y si fuera falsa sería cierta, eso quiere decir que no tenemos un concepto claro y bien definido de "cierta" o "falsa", si lo tuviéramos nos daría lugar a una paradoja.

La ventaja de la lógica proposicional es que nos permite construir proposiciones compuestas a partir de otras más simples que ya conozcamos. Por ejemplo, si P y Q son dos proposiciones conocidas, entonces la oración "P es cierta y Q también es cierta" también es una proposición. En efecto, como P y Q son proposiciones, sabemos que significa que sean ciertas (o falsas). Que nuestra nueva proposición sea cierta significa exactamente que tanto P como Q sean ciertas, y que sea falsa quiere decir exactamente que al menos una de las dos es falsa.
Lo que acabamos de hacer, indirectamente, es definir un conector lógico. Vamos a detallar esto:

Definición. Si n es un numero natural positivo, un conector lógico $n-ario\ C$ es una regla bien definida que nos permite conectar n proposiciones conocidas $P_1, P_2, ..., P_n$ para formar una nueva proposición $C(P_1, P_2, ..., P_n)$.

Cuando hablamos de una regla "bien definida" nos referimos a que a partir de los valores de verdad de las proposiciones $P_1, P_2, ..., P_n$ debemos ser capaces de conocer el valor de verdad de $C(P_1, P_2, ..., P_n)$. Para hacer esto habitualmente definimos los conectores lógicos mediante lo que se conoce como su tabla de verdad. Para entender mejor lo que significa esto vamos a dar algunos ejemplos de conectores lógicos:

Ejemplo. Estos son algunos de los conectores lógicos más importantes en la lógica proposicional:

- Dijimos anteriormente que habíamos definido un conector lógico de manera implícita. No es otro que el conector lógico "y", también llamado "conjunción lógica", un conector binario con la siguiente tabla de verdad:

P	**Q**	**P y Q**
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Falso
Falso	Verdadero	Falso
Falso	Falso	Falso

  En cada columna recogemos los valores de verdad de cada proposición. El valor de verdad de la columna de la derecha (el de "P y Q") depende de los valores de verdad del resto de columnas. Si incluimos todas las combinaciones posibles de valores de verdad de P y de Q, estamos definiendo sin lugar a errores el valor de verdad de P y Q en función de los valores de verdad de P y de Q. Este es exactamente el requisito que pusimos para que un conector lógico estuviera bien definido, así que vemos que las tablas de verdad son suficientes para definir correctamente un conector lógico.
  Podemos generalizar el conector "y" a un conector $n - ario$, que sea cierto cuando todos sus argumentos sean ciertos, y falso en caso contrario.

- Ahora vamos a introducir un conector unario muy importante, conocido como "negación". A partir de una proposición P, construimos una proposición "no P", cuyo valor de verdad es siempre el opuesto del de P. Esto lo podemos recoger en una tabla de verdad:

P	**no P**
Verdadero	Falso
Falso	Verdadero

- Otro conector lógico binario muy importante es la disyunción lógica, o conector

"o", que recogemos en la siguiente tabla de verdad:

P	**Q**	**P o Q**
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Verdadero
Falso	Verdadero	Verdadero
Falso	Falso	Falso

Al igual que hicimos con la conjunción, podemos generalizarlo a un conector $n - ario$, que sea falso cuando todos sus argumentos sean falsos, y cierto en caso contrario.

- A pesar de que en el lenguaje corriente el conector "o" se utiliza con un significado excluyente (es decir, si una persona dice "voy a ir al cine o al teatro", no considera la posibilidad de ir a ambos sitios), en lógica proposicional esto no es así. Sin embargo, si que existe ese conector lógico, que se conoce como "o exclusivo", y tiene la siguiente tabla de verdad:

P	**Q**	**o bien P o bien Q**
Verdadero	Verdadero	Falso
Verdadero	Falso	Verdadero
Falso	Verdadero	Verdadero
Falso	Falso	Falso

Es decir, "o bien P o bien Q" es cierto cuando entre las proposiciones P y Q hay *exactamente* una que es verdadera.

- El último conector lógico que vamos a introducir es el de implicación, que recoge la idea de consecuencia lógica:

P	**Q**	**P implica Q**
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Falso
Falso	Verdadero	Verdadero
Falso	Falso	Verdadero

Hay que tener cuidado con la idea de "consecuencia lógica", pues puede chocar con la intuición. Según esta tabla de verdad una proposición falsa implica cualquier otra, algo bastante aceptado y comprendido entre los matemáticos, pero un poco contraintuitivo. Por ejemplo, si la proposición P es "hoy ha llovido" y la proposición Q es "yo soy el Papa", entonces (si suponemos que hoy no ha llovido) la proposición "P implica Q" es cierta. Traduciendo las proposiciones y conectores los lógicos al lenguaje coloquial, estamos afirmando que la oración "Si hoy ha llovido, entonces yo soy el Papa" es cierta, lo cual suena absurdo, aunque en realidad no lo es.

- Podríamos definir de manera explícita conectores $n - arios$ para $n > 2$ mediante tablas de verdad, pero no es necesario, puesto que cualquiera de ellos se puede expresar como combinación de los conectores que ya hemos definido. Un ejemplo de cómo se podría hacer esto sería definir el conector cuaternario C diciendo que la proposición $C(P, Q, R, S)$ es cierta exactamente cuando sea cierta la proposición "(P o Q) implica (R y S)", cuyo valor de verdad podemos conocer a partir de las tablas de verdad de los conectores lógicos que ya hemos definido.

3.1.2. El proyecto de Scratch

Al igual que en el resto de actividades, el proyecto de Scratch correspondiente a esta actividad está compartido en la página de Scratch, para que sea accesible por todo el mundo. Además, existe la posibilidad de tomar el proyecto y reinventarlo, haciendo muy sencillo incluir nuevos acertijos, u otros distintos. El enlace es `https://scratch.mit.edu/projects/702700836/`.

El funcionamiento del programa es relativamente sencillo, simplemente contiene unas instrucciones de uso y va recorriendo los siete acertijos en orden, registrando las respuestas que el alumno da a cada una, permitiendo varios intentos hasta que se acierte, y anotando el número de fallos en el proceso. Ahora vamos a mostrar los siete acertijos, comentando el enunciado, la respuesta, y los conceptos que se trabajan en cada caso:

1. En el primer acertijo la pista que nos da Porcia es que de las tres inscripciones, a lo sumo una es cierta, y estas muestran las siguientes afirmaciones:

Podemos deducir que el retrato está en el cofre de plata, porque como la proposición inscrita en el cofre de oro y el de plomo son opuestas (una y solo una de ellas será cierta), la proposición inscrita en el cofre de plata debe ser falsa, puesto que si no habría dos inscripciones ciertas, en contra de la hipótesis. Por lo tanto, el retrato está en el cofre de plata.

En esta prueba trabajamos el concepto de afirmaciones opuestas, y reafirmamos (implícitamente) el hecho de que la proposición "P o no P" siempre es verdadera.

2. En este caso Porcia nos dice que entre las inscripciones hay al menos una proposición cierta, y al menos una proposición falsa:

Podemos deducir que el retrato está en el cofre de oro. Si estuviera en el de plomo, los tres enunciados serían ciertos, mientras que si estuviera en el de plata serían los tres falsos. Por lo tanto, por descarte, debe encontrarse en el de oro, y entonces podemos comprobar que, de hecho, las dos primeras inscripciones son ciertas, y la última es falsa.

Con esta actividad introducimos el recurso de separar casos, bastante útil tanto en este tipo de acertijos como en las matemáticas en general.

3. En esta prueba cada inscripción tiene dos afirmaciones, y Porcia nos explica que ninguna inscripción tiene dos afirmaciones falsas.

Si el retrato estuviera en el cofre de plomo sus dos afirmaciones serían falsas,

así que debe estar o bien en el de oro o bien en el de plata. Ahora podemos fijarnos en el hecho de que la primera afirmación de cada uno de estos cofres es equivalente, mientras que las segundas son incompatibles (no pueden ser ambas ciertas a la vez). Por lo tanto, no puede ser que las primeras afirmaciones sean falsas, pues entonces uno de los dos cofres tendría dos enunciados falsos. Deducimos, pues, que el retrato no está en el cofre de oro, está en el de plata. Nótese que en esta actividad está presente de manera implícita (nosotros podemos explicitársela a los alumnos para que se den cuenta) el conector "o", puesto que al fin y al cabo la pista que nos ha dado Porcia es que en todos los cofres la proposición "(1) o (2)" es verdadera.

4. Esta prueba es como la anterior en el hecho de que cada inscripción tiene dos proposiciones. Pero esta vez la pista es que en un cofre hay dos afirmaciones falsas, en otro dos verdaderas, y en otro hay una falsa y una verdadera:

Si el retrato estuviera en el cofre de oro, entonces tanto el cofre de oro como el de plata tendrían dos enunciados falsos, lo cual va en contra de nuestras hipótesis. Si el retrato estuviera en el cofre de plata, entonces tanto el cofre de plomo como el de plata tendrían una de sus proposiciones falsas y la otra verdadera, lo cual también es incompatible con nuestras hipótesis. Por lo tanto, la única posibilidad es que el retrato esté en el cofre de plomo (y comprobamos que en ese caso el cofre de plata tiene dos enunciados verdaderos, el de plomo

dos falsos, y el de oro uno de cada).

5. En las tres últimas pruebas Porcia no nos va a dar información sobre la veracidad de las inscripciones explícitamente, esta nos vendrá de otra manera más indirecta:

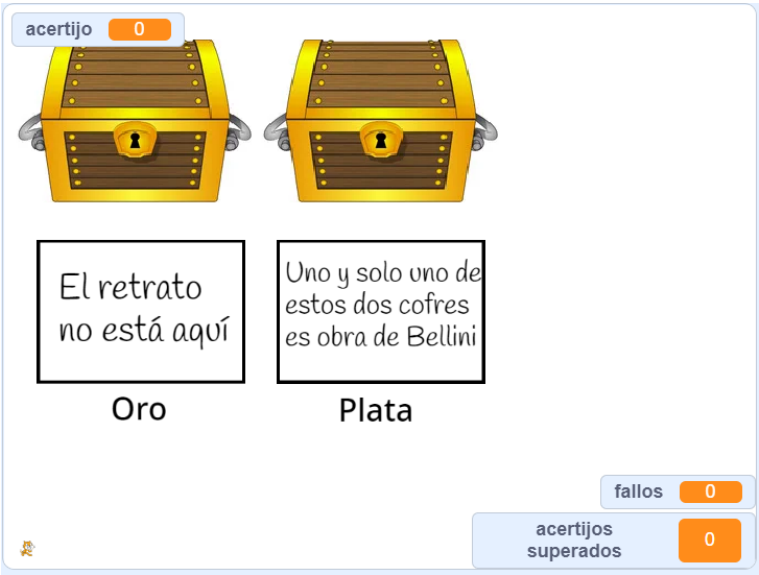
   Además, este quinto acertijo también es distinto a los demás por otro motivo, y es que el cofre que no está vacío no contiene un retrato, sino que contiene una daga. En este caso el pretendiente debe evitar la daga si quiere tener la posibilidad de casarse con Porcia:

Centremos nuestra atención en el cofre de plomo. Si éste es cierto, entonces lo ha hecho Bellini, por lo tanto es el único cofre que ha hecho Bellini, luego la inscripción del cofre de oro y la del de plata son falsas. En consecuencia, el de oro nos dice que no contiene la daga, y el de plata nos dice que contiene la daga (vemos que en este caso las tres inscripciones son compatibles, no nos llevan a ninguna contradicción, esto es una posibilidad real). Por otro lado, si el cofre de plomo lo ha hecho Cellini entonces es el único cofre que puede ser falso, los otros dos los debe haber hecho Bellini, así que la daga está en el cofre de oro. De nuevo, comprobamos que las tres inscripciones de los cofres son compatibles, así que esta también es una posibilidad real. En cualquiera de los dos casos (y no puede haber más, pues el cofre de plomo solo puede ser cierto o falso), el cofre de plomo no contiene la daga, así que este es el

que deberán seleccionar nuestros alumnos, es el único que se considera como respuesta correcta en el programa.

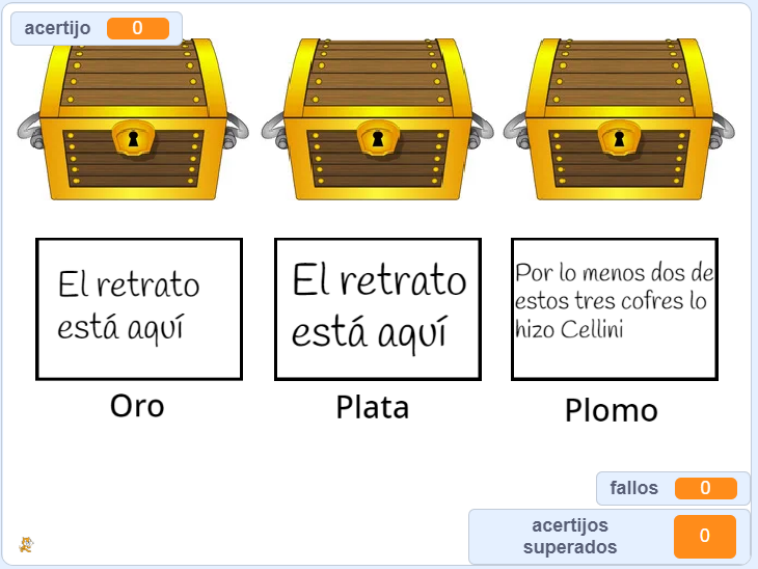
Observamos que en este acertijo hay un cierto incremento de dificultad: ahora no podemos simplemente separar casos, pues la posición de la daga es indeterminada, y además para resolverlo tendremos que negar una proposición no trivial (la inscripción del cofre de plomo).

6. Volvemos al caso en que uno de los cofres contiene un retrato que hay que encontrar, y todos han sido inscritos por Bellini o Cellini. Aunque en este acertijo solo hay dos cofres:



De manera análoga al anterior acertijo, la mejor estrategia es discutir que ocurriría si el cofre de plata lo hubiera forjado Bellini o Cellini. Si es obra de Bellini, entonces el cofre de oro es obra de Cellini, luego contiene el retrato. Si el cofre de plata es obra de Cellini entonces el de oro también debe serlo, pues no puede haber exactamente un cofre que haya hecho Bellini. La conclusión es la misma que en el otro caso, el retrato está en el cofre de oro.

7. En este acertijo también sabemos que todos los cofres son obra de Bellini o Cellini:



Volvemos a analizar la autoría del cofre de la derecha. No puede ser obra de Cellini, porque si lo fuera entonces sería el único cofre con una inscripción falsa, y el retrato estaría tanto en el cofre de oro como en el de plata. Por lo tanto, la única posibilidad es que el cofre de plomo sea de Bellini, lo cual fuerza a que el cofre de oro y el de plata deben ser de Cellini, y de ahí deducimos que el retrato está en el cofre de plomo.

Tanto en este acertijo como en el anterior seguimos reforzando la habilidad de negar proposiciones, es crucial en las pruebas de Bellini y Cellini para llegar a la respuesta correcta.

3.1.3. Planteamiento

Esta actividad está pensada para una clase de cuarto de la ESO, de la asignatura de Matemáticas Orientadas a las Enseñanzas Académicas. Puede ser una propuesta entretenida y educativa para algún momento del curso con una menor exigencia (por ejemplo, al final de curso, cuando las notas ya están puestas), o para algún día de la semana en el que los alumnos suelan tener más problemas para concentrarse, como un viernes, o un día en el que la clase de matemáticas sea a última hora.

La resolución de los acertijos que se encuentran en el proyecto de Scratch debe ir precedida por una clase un poco más teórica, en la que los alumnos aprendan los

conceptos básicos sobre la lógica proposicional, y se acostumbren un poco a las inferencias más clásicas, para que puedan extraer conclusiones de unas hipótesis que reciban.

Después de esto estarán preparados para dedicar una sesión de clase en el aula de informática resolviendo los acertijos, separados en pequeños grupos, para aumentar la motivación, introduciendo una pequeña competición entre la clase.

3.1.4. Metodología

En el desarrollo de esta actividad utilizaremos principalmente tres metodologías:

- Lección magistral participativa. Para exponer de una manera rápida y efectiva los contenidos sobre lógica proposicional, esta actividad comenzará con una sesión de clase en la que se utilizará el método expositivo. Esto tiene como ventaja que podremos detectar y resolver de manera inmediata las dudas que tengan los alumnos sobre la materia.
  La idea es dedicar el mínimo tiempo posible a definir formalmente los conceptos a estudiar, para disponer del mayor tiempo posible para resolver problemas e interiorizar los conceptos más importantes.

- Aprendizaje cooperativo. Para resolver los siete acertijos en el aula de informática, el profesor dividirá a los alumnos en grupos de tres personas, adaptándolos a las necesidades de los alumnos. El trabajo cooperativo permite un intercambio de ideas más rico que el pensamiento individual, y además hace más difícil perder la motivación al quedarse atascado en un problema, es más difícil quedarse atascado entre tres personas que uno solo. A mayores, podemos mencionar que es más sencillo detectar errores en razonamientos ajenos que en los propios, por lo que de esta manera los alumnos pueden ayudarse comprobando si sus respuestas son correctas de una manera más efectiva que si estuvieran solos.

- La resolución de problemas. Tanto en la clase de lógica como en el aula de informática el profesor debe tener un rol activo en la resolución de problemas,

ya que son la parte central de esta actividad. La labor del docente no es simplemente la de plantear los problemas y observar a los alumnos, ni tampoco la de resolverlos él mismo mientras los alumnos "miran y aprenden", sino que su papel es el de un apoyo, debe hacerle a los alumnos las preguntas adecuadas para que puedan progresar adecuadamente en la resolución de los acertijos.

3.1.5. Temporalización

Como ya se ha comentado, la actividad se desarrollará a lo largo de dos sesiones de clase, que se suponen de la duración típica en la ESO: 50 minutos. Por otro lado, también hemos comentado que, al estar relacionada con el bloque de contenidos comunes, su colocación al lo largo del curso no es crucial, puede funcionar bien como nexo de unión entre dos unidades didácticas, o al inicio o final de una evaluación.

3.1.6. Evaluación

Para evaluar esta actividad utilizaremos distintos instrumentos:

1. La observación:

   Es importante que los alumnos muestren un alto nivel de participación en la actividad, tanto en la lección magistral como, sobretodo, en la realización de los acertijos en grupo. Se valorará no solo el nivel de atención y desempeño de los alumnos a lo largo de las dos sesiones, sino también el nivel de progreso que sean capaces de alcanzar, la colaboración con sus compañeros a la hora de trabajar en grupo, y el hecho de que tengan un comportamiento que favorezca un buen clima de aula.

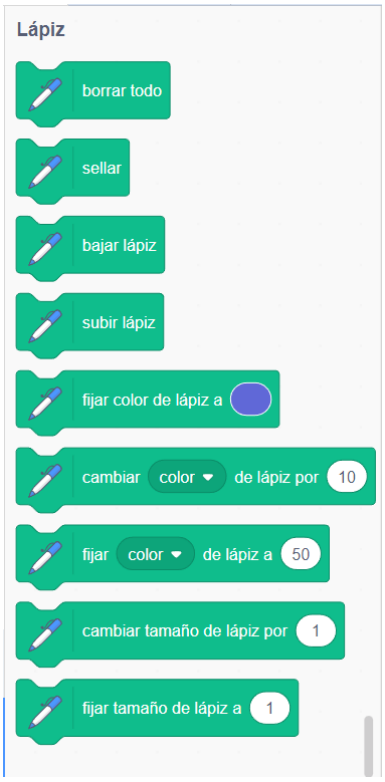
2. El desempeño en los acertijos:

   Se busca que los alumnos lleguen a la respuesta adecuada mediante un proceso adecuado, así que no valoraremos tan solo la puntuación que obtengan (es decir, el número de acertijos resueltos y errores cometidos), se tendrá en cuenta el razonamiento seguido por los alumnos para decidir que cofre abrir. Para conseguir esto se puede recurrir a dos métodos distintos, en función del tiempo que se disponga y la velocidad a la que estén resolviendo los alumnos los

acertijos. Si los grupos van bien de tiempo se les puede pedir que después de contestar correctamente a un acertijo un miembro del equipo se encargue de poner por escrito el razonamiento que han seguido para deducir la respuesta. Si el tiempo es algo más ajustado, el profesor puede preguntarles a los alumnos que le expliquen de palabra alguno de los razonamientos seguidos.

3.2. Actividad 2: Geometría con Scratch

En esta actividad vamos a hacer uso de una extensión de Scratch para dibujar y estudiar las figuras planas a un nivel de primero de la ESO. Como ya mencionamos anteriormente, las extensiones de Scratch nos añaden bloques que incluyen nuevas funcionalidades. En el caso del lápiz, nos añade la posibilidad de dibujar el "rastro" que deja la trayectoria de un objeto. Estos son los nuevos bloques que podremos utilizar:



La idea es que cada objeto tiene asociado un lápiz, y cuando este baja, dibuja una línea que pasa por donde pase nuestro objeto. También podemos utilizar la opción de "sellar" para dejar marcado un dibujo de nuestro objeto en un momento específico,

captando su posición, dirección y tamaño. Se pueden modificar ciertas propiedades del lápiz, como el color, grosor, transparencia, saturación y brillo.
Podemos aprovechar esta extensión para dibujar una gran variedad de figuras planas, si sabemos cómo hacer que un objeto se mueva describiendo una cierta trayectoria. Por ejemplo, si queremos dibujar un triángulo equilátero de lado 10 lo podemos hacer con tan solo 6 comandos: bajar el lápiz, avanzar 10 pasos, girar 120°, avanzar 10 pasos, girar 120°, avanzar 10 pasos. De una manera similar se pueden representar otros polígonos regulares, y con técnicas algo más complejas se pueden representar todos los polígonos que queramos. Nuestro objetivo es aprovechar estas funcionalidades para crear una actividad para una clase de matemáticas de primero de la ESO en la que se repasen los conceptos relativos a la unidad didáctica sobre figuras planas de una manera visual y divertida.

3.2.1. Contenidos

En este unidad nos centramos principalmente en los contenidos relativos al Bloque 3, de Geometría, como por ejemplo:

- *Ángulos y sus relaciones.*
  Trabajaremos el concepto de ángulo al hablar de los ángulos interiores y exteriores de un polígono, así como el concepto de ángulo suplementario, a la hora de relacionar el ángulo interior de un polígono y el ángulo que debe girar nuestro objeto para dibujar el polígono en Scratch.

- *Figuras planas elementales: triángulo, cuadrado, figuras poligonales.*
  En esta actividad se trabajará con todos estos polígonos, junto con sus propiedades más importantes.

- *Clasificación de triángulos. Rectas y puntos notables del triángulo. Uso de medios informáticos para analizarlos y construirlos.*

- *Medida y cálculo de ángulos de figuras planas. Cálculo de áreas y perímetros de figuras planas. Cálculo de áreas por descomposición en figuras simples.*
  Tanto en la demostración de la fórmula genera del área de un polígono regular,

como en el cálculo de la suma de ángulos interiores de un polígono convexo, se debe realizar una descomposición de la figura en triángulos.

- *Uso de herramientas informáticas para estudiar formas, configuraciones y relaciones geométricas.*
  Este es el objetivo principal de la actividad, el uso de medios informáticos para trabajar con propiedades geométricas de las figuras planas.

Al igual que en la actividad anterior, también nos centramos en los contenidos comunes del Bloque 1:

- *Utilización de medios tecnológicos en el proceso de aprendizaje para facilitar la comprensión de propiedades geométricas o funcionales y la realización de cálculos de tipo numérico, algebraico o estadístico.*

Ahora incluimos un desarrollo teórico de los contenidos principales que vamos a tratar, para el que nos hemos apoyado en los apuntes de Marea Verde (2022):

Definición. Un punto es una figura geométrica sin dimensión (no tiene longitud, altura ni anchura) que determina una posición en el plano.
Una recta es una línea que se extiende infinitamente en una misma dirección, abarcando todos los (infinitos) puntos que hay en esta.
Una semirrecta es la porción de una recta que queda a un lado de un punto de ésta. A este punto se le llama "origen de la semirrecta".
Un segmento es el conjunto de puntos de una recta comprendido entre dos puntos que estén en ella. Los segmentos son la intersección de dos semirrectas. Si tenemos dos puntos A, B de una misma recta, designamos por $\overline{AB}$ al segmento que definen, y lo llamaremos "segmento de extremos A y B".

Definición. Un ángulo es una región del plano que queda limitada por dos semirrectas con un origen común.
Si estas dos semirrectas son la misma, entonces lo llamamos ángulo completo, y decimos que mide 360°.
Si las dos semirrectas forman una recta, entonces el ángulo que forman es de la mitad que un ángulo completo, 180°, y lo llamamos ángulo llano.

Al ángulo que es la mitad de un ángulo llano, 90°, lo llamamos ángulo recto.
Las dos semirrectas que limitan un ángulo se llaman lados, y el origen común a las dos semirrectas se llama vértice. Si tenemos dos semirrectas A, B con un origen común O, entonces podemos denotar al ángulo que determinan por $A\hat{O}B$.

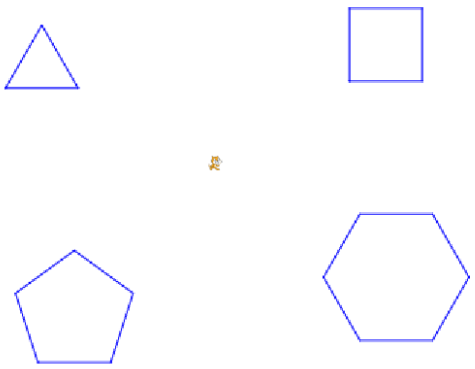
Definición. Podemos clasificar los ángulos según su amplitud. Un ángulo mayor que uno llano se dice cóncavo, y uno que es menor es convexo. A su vez, podemos clasificar los ángulos convexos en agudos (menores que un ángulo recto), rectos, y obtusos (mayores que un ángulo recto).
Llamamos ángulos consecutivos a dos ángulos con un mismo vértice y un lado común. Si tenemos dos ángulos consecutivos en los que los lados no comunes forman un ángulo llano, decimos que son ángulos adyacentes o suplementarios. Si forman un ángulo recto decimos que son ángulos complementarios.
Dos ángulos son opuestos por el vértice si tienen el mismo vértice y los lados de uno son semirrectas opuestas a los lados del otro.

Definición. Definimos una línea poligonal como una sucesión finita de segmentos (en la cual dos segmentos consecutivos no están alineados), donde dos segmentos distintos tienen intersección vacía o coinciden en un único punto, que es un extremo de cada uno de los segmentos. Una línea poligonal es cerrada si su último segmento interseca con el primero, y abierta en caso contrario
Un polígono es una región del plano acotada que queda delimitada por una linea poligonal cerrada.
Los lados de un polígono son los segmentos que forman la línea poligonal que lo delimita. Sus vértices son los puntos que obtenemos al intersecar dos lados consecutivos. Las diagonales son los segmentos que unen vértices no consecutivos (dos vértices son consecutivos si están unidos por un lado).
Llamamos ángulo interior de un polígono al ángulo formado por dos lados consecutivos, y ángulo exterior a su ángulo suplementario.
Podemos clasificar a los polígonos en función de sus ángulos interiores: si un polígono tiene todos sus ángulos interiores convexos, decimos que es un polígono convexo, y si alguno de ellos es cóncavo, entonces es un polígono cóncavo.

Ejemplo. También podemos clasificar a los polígonos por su número de lados:

- El menor número de lados que puede tener un polígono es 3. Y en ese caso diremos que es un triángulo.
- Los polígonos de cuatro lados se llaman cuadriláteros.
- Los de cinco pentágonos
- Los de seis hexágonos

Y así sucesivamente, uno de siete es un heptágono, de ocho un octógono, etc. En la siguiente imagen podemos ver un triángulo, un cuadrilátero, un pentágono y un hexágono, dibujados en Scratch:



Los polígonos de la imagen anterior no eran polígonos cualesquiera, tienen una propiedad muy interesante que capturamos en la siguiente definición:

Definición. Si un polígono tiene todos sus lados iguales se dice que es equilátero. Un polígono cuyos ángulos interiores son iguales decimos que es equiángulo. Los polígonos que son equiláteros y equiángulos son los polígonos regulares. Un polígono que no es regular decimos que es irregular.
Los polígonos regulares tienen propiedades especiales, que nos permiten definir los siguientes conceptos, que tienen validez solo para polígonos regulares:

- El centro es el punto que equidista de todos los vértices.
- Los radios son los segmentos que unen el centro con los vértices del polígono.

- Los ángulos centrales son los ángulos determinados por dos radios consecutivos y el centro del polígono.
- Una apotema es un segmento que une el centro del polígono con el punto medio de uno de los lados.

Teorema. El área de un polígono regular de n lados de longitud l y apotema de longitud Ap es de $nl\frac{Ap}{2}$. Al número nl también se le llama perímetro, es la suma de las longitudes de los segmentos que componen al polígono. Si lo denotamos por p, podemos reescribir el valor del área como $\frac{pAp}{2}$.

Teorema. La suma de todos los ángulos interiores de un polígono convexo de n lados es de $(n-2)180°$.
En particular, la suma de los ángulos interiores de un triángulo es de $180°$, un resultado muy conocido.

Corolario. En un polígono regular de n lados, los ángulos interiores miden $\frac{n-2}{n}180°$.

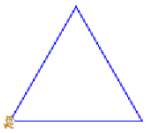
Demostración. La suma de todos los ángulos interiores de un polígono de n lados es, por el teorema anterior, $(n-2)180°$, y como hay n de ellos y son todos iguales, cada uno debe medir $\frac{n-2}{n}180°$. □

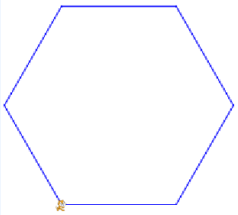
3.2.2. El proyecto de Scratch

En este caso, a diferencia de la actividad anterior, no habrá un único proyecto de Scratch que deban resolver todos los alumnos, sino que el objetivo es que sean ellos quienes programen algunos algoritmos para dibujar figuras geométricas. Presentaremos ahora el aspecto que podrían tener algunos de estos programas, junto con el dibujo que nos sacan por pantalla. Hemos compartido estos algoritmos en un proyecto de Scratch, que se puede visitar en `https://scratch.mit.edu/projects/704207913/`.

Aquí podemos ver uno de los programas más sencillos, que es el que nos permite dibujar algunos polígonos regulares. Para ello solo tenemos que movernos una cierta distancia (la longitud de nuestro lado), y girar cierto ángulo (el ángulo exterior de

nuestro polígono). Después, repetimos esta sentencia tantas veces como lados tenga nuestro polígono y obtenemos la figura que queríamos. Debajo vemos el aspecto que tiene el algoritmo para un triángulo y un hexágono:

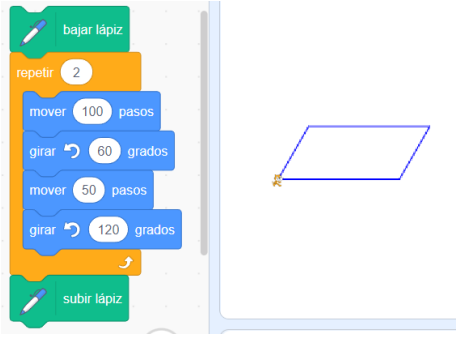




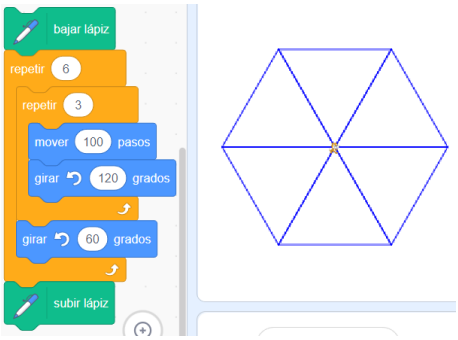
No es difícil, a partir de los algoritmos anteriores, deducir como es uno que puede dibujar un polígono genérico, de cualquier numero de lados:

A partir de este podemos dibujar un polígono con un gran número de lados, para que los alumnos vean como se aproxima a una circunferencia.

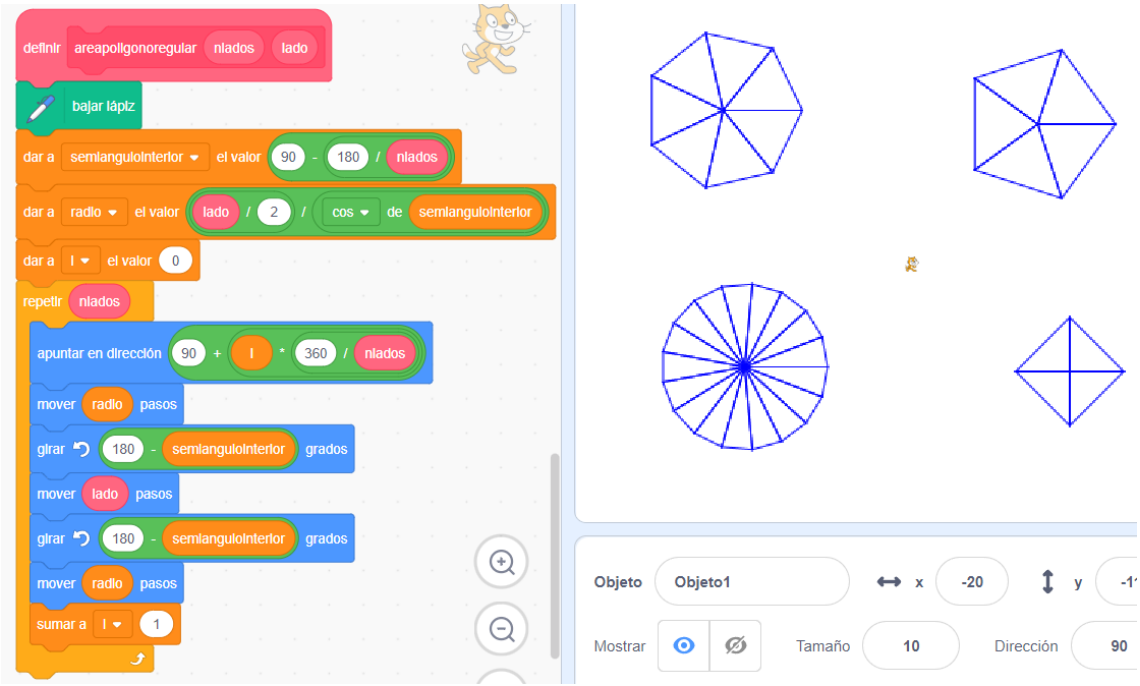
Mediante el siguiente algoritmo podemos dibujar cualquier paralelogramo (un cuadrilátero con lados opuestos paralelos), tomando como argumentos las medidas de los lados y los ángulos interiores:



Es conocida la construcción de un hexágono regular a partir de 6 triángulos regulares. Como vemos, no es difícil de hacer, los alumnos podrían llegar a desarrollar este algoritmo por ellos mismos:



A modo de curiosidad, nosotros podemos generalizar este algoritmo para obtener uno que nos dibuje un n-ágono regular a partir de n triángulos isósceles idénticos:



En la imagen podemos ver cómo dibuja polígonos de siete, cinco, diecisete y cuatro lados mediante una malla de triángulos apropiada. Si bien no está al alcance de los alumnos programar este algoritmo por su cuenta, nosotros podemos utilizarlo para mostrarles una visualización de la fórmula general del área de un polígono regular,

puesto que en nuestra imagen vemos fácilmente que el polígono está formado por n triángulos iguales, cada uno de área $\frac{lado \cdot Apotema}{2}$.

Aunque no hay un algoritmo concreto que esperemos sobre esta actividad, también podemos utilizar Scratch para visualizar la prueba de que los ángulos de un polígono convexo de n lados suman $(n-2)180°$, dibujando polígonos de manera "progresiva", añadiendo triángulos progresivamente cada vez que queramos añadir un lado, como podemos ver en la siguiente imagen:

En este dibujo hecho con Scratch podemos ver un polígono de seis lados compuesto por cuatro triángulos, de manera que deducimos que la suma de sus ángulos es la suma de los ángulos de esos triángulos (ya que todos los ángulos de los triángulos forman parte de ángulos interiores de nuestro polígono), o sea 720°.

Otra forma de entender este resultado es mediante el programa anterior. Si queremos sumar todos los ángulos interiores de un polígono (regular) de n lados, nos fijamos en que queda dibujado mediante n triángulos. Los ángulos de todos estos triángulos suman n·180°. Sin embargo, los ángulos desiguales de esos triángulos no son parte de los ángulos interiores de nuestro polígono original. Por lo tanto, si restamos a n·180°la suma conjunta de todos estos ángulos (que es un ángulo completo, de 360°) obtenemos la suma de los ángulos interiores de nuestro polígono de n lados, que es de (n-2)180°.

3.2.3. Planteamiento

Esta actividad está pensada para una clase de matemáticas de primero de la ESO. El momento ideal del curso para plantearla creemos que es al final de la unidad didáctica correspondiente a las figuras planas, aprovechándola como un repaso de algunos de los conceptos centrales de esta unidad.

Antes de proponer a los alumnos la parte de la actividad consistente en dibujar en Scratch, es necesario que tengan una cierta familiaridad con Scratch, por lo que debemos realizar una sesión de clase en la que les enseñemos los elementos más

básicos: bloques, objetos, el escenario, el lápiz, y unas nociones básicas sobre el bloque "repetir", utilizado para realizar bucles.
Después de esta sesión de "tutorial", los alumnos ya están listos para pasar una clase tratando de estudiar las propiedades geométricas de los polígonos utilizando Scratch para ello.

3.2.4. Metodología

En esta actividad buscamos que los alumnos aprendan por descubrimiento, que prueben cosas ellos mismos para descubrir como llegar a la solución correcta, y que puedan experimentar por su cuenta las propiedades de los objetos geométricos que estamos tratando. Es por eso que nuestras metodologías se basan en una participación activa por parte de los alumnos:

- Aprendizaje basado en tareas:
  En el aprendizaje basado en tareas, o enfoque por tareas, se intenta centrar la enseñanza en el estudiante. Una tarea es una secuencia organizada de tal forma que ayude a los alumnos a lograr la realización de un actividad compleja. Durante este proceso se trata que el alumno sea progresivamente responsable de su aprendizaje, a partir de la resolución de problemas propios, lo que facilita la motivación y permite un aprendizaje significativo.
  Hemos elegido esta metodología porque acerca a los estudiantes a lo que son en realidad las matemáticas, al trabajo que hace un matemático: resolver problemas por su cuenta, con un enfoque creativo, a través del razonamiento.

- Aprendizaje experiencial:
  El aprendizaje experiencial es un tipo de aprendizaje práctico y activo basado en la idea de que la educación debe ser una constante reconstrucción de la experiencia de lo que se intenta aprender. En este caso, se trata de que los alumnos tengan una experiencia que se asimile lo mayor posible a la programación, con un objetivo de modelizar un fenómeno matemático mediante un software informático.

3.2.5. Temporalización

La actividad está pensada para tener una duración de dos sesiones de 50 minutos, situadas en un aula de informática, o alternativamente disponiendo de al menos un teléfono o tablet para cada alumno, para que tengan todos ellos acceso a Scratch.

1. La primera sesión está pensada como una en la que los alumnos deben familiarizarse con el entorno de Scratch, aprender los comandos básicos, y las funcionalidades del lápiz. La metodología debe ser activa, los alumnos deben hacer la mayor parte del trabajo, experimentando por su cuenta con las posibilidades que ofrece el programa, mientras que el profesor debe ser tan solo una guía, su objetivo es el de dar indicaciones generales a los alumnos y resolver sus dudas cuando sea pertinente.

2. En la segunda sesión trabajaremos los conceptos de geometría propiamente dicha. Al igual que en la anterior sesión, el trabajo del profesor es tan solo el de guiar a los alumnos, quienes deben experimentar por su cuenta. Algunas de las indicaciones que puede darle el profesor a los alumnos son las siguientes:

    - Tratar de dibujar polígonos regulares, y calcular su perímetro y área cuando sea posible.
    - Hacer un programa que dibuje cualquier polígono regular, si se lo pedimos adecuadamente. ¿Qué pasa si el número de lados es muy grande?
    - ¿Puedes hacer un programa que dibuje cuadriláteros que no sean cuadrados? ¿Y puedes hacerlo sin que tampoco sean rectángulos?
    - Se les puede mostrar un hexágono regular construido a partir de triángulos regulares, y pedirles que lo repliquen.
    - A partir de imágenes de polígonos regulares dibujados a partir de triángulos, les podemos pedir que reflexionen sobre la fórmula del área general, o sobre la suma de los ángulos interiores de un polígono convexo.

Como ya hemos comentado, la actividad se presenta más naturalmente al finalizar la unidad didáctica correspondiente a las figuras planas, una vez los alumnos tienen ciertos conocimientos sobre los polígonos que van a representar.

3.2.6. Evaluación

La forma de evaluar es análoga a la de la actividad anterior, estaremos utilizando la observación para medir el nivel de participación y esfuerzo de los estudiantes en la actividad, y de la misma manera valoraremos su desempeño en la tarea. Para conseguir esto último, recogeremos el proyecto de Scratch de cada alumno, y valoraremos la creatividad, la limpieza del código (presencia de comentarios con explicaciones, organización de los bloques, etc) y la cantidad y calidad de las figuras que sean capaces de dibujar.

3.3. Actividad 3: Números primos

Esta actividad es, de entre las tres que proponemos, la más cercana a la programación informática, una de las principales aplicaciones didácticas de Scratch. A pesar de ello, hemos aprovechado la programación como recurso para la enseñanza de la aritmética de los números naturales, para adaptarnos al currículo de matemáticas de segundo de la ESO. El objetivo es que los estudiantes sean capaces, con la ayuda del profesor, de escribir un algoritmo en Scratch que tome como entrada un número natural positivo, y que compruebe si ese número es primo o no.

Si bien esta actividad no es sencilla (teniendo en cuenta que la mayoría de los alumnos nunca habrán estudiado nada relacionado con la programación), creemos que es posible que todos los alumnos lleguen a participar plenamente en ella, y con la ayuda del profesor y el trabajo en equipo sean capaces de desarrollar el programa que se pide. Es importante que no todos los problemas a los que se enfrenten los alumnos sean "problemas tipo", pues de esta manera favorecemos el pensamiento creativo y la búsqueda de soluciones a problemas completamente desconocidos, facultades fundamentales en las matemáticas.

3.3.1. Contenidos

En esta actividad nos centramos principalmente en los contenidos del Bloque 2, de Números y Álgebra, como por ejemplo:

- *Divisibilidad de los números naturales. Criterios de divisibilidad.*
  El concepto de divisibilidad en si mismo es necesario para comprender la noción de número primo. Utilizamos los criterios de divisibilidad en un sentido genérico, entendiendo que un número es múltiplo de otro si al realizar la división entera del primero por el segundo el resto es cero. En los criterios de divisibilidad, al fin y al cabo lo que hacemos es calcular de manera eficiente el resto de una división entera.

- *Números primos y compuestos. Descomposición de un número en factores primos.*
  Estos son los contenidos centrales de la actividad: el concepto de número primo, y la forma en que comprobamos si un número es primo o no (tratando de descomponerlo como producto de dos números naturales.)

Ahora incluimos un desarrollo teórico de los contenidos principales que vamos a tratar, para el que nos hemos apoyado en los apuntes de Marea Verde (2022):

Definición. Se dice que un número natural m es múltiplo de otro, n, si existe un cierto k natural tal que $m = k \cdot n$. De esta manera, los múltiplos de un número n son aquellos que se escriben como un cierto numero natural multiplicado por n. Se denota al conjunto de los múltiplos de n como (n), y se le llama el ideal generado por n. Es decir, $(n) = \{n, 2n, 3n, 4n, ...\}$.
Como podemos observar, se trata de un conjunto infinito.

Definición. Un número natural n es divisor de otro, m, si m es un múltiplo de n. Observamos que esta relación es exactamente la opuesta a la multiplicidad, es decir: m es múltiplo de n si, y solo si, n es divisor de m. Denotamos al conjunto de los divisores de un número natural n como $d(n)$. Si un número es divisor de otro, entonces debe ser menor, por lo que $d(n) \subset \{1, 2, ..., n\}$.

Observación. Para cada $n \in \mathbb{N}$, siempre se cumple que $1, n \in d(n)$. O sea, todo número natural es divisor de si mismo, y 1 es un divisor de todos los números naturales.

Teorema. Si m, n son dos números naturales, entonces n es un divisor de m si, y solo si, el resto de la división entera de m entre n es 0.

Demostración. Si n es un divisor de m, entonces $m = kn$ para un cierto k natural. Nos fijamos en que $m = kn$ es una división entera de m entre n, y como la división es única, el resto es 0.
Recíprocamente, si el resto de la división de m entre n es 0, entonces $m = kn$ para un cierto $k \in \mathbb{N}$, luego n es un divisor de m. □

Este teorema nos dice que si queremos comprobar si un número natural es múltiplo de otro no necesitamos escribirlo de manera explícita como un múltiplo suyo, nos basta con calcular el resto de la división entera y comprobar que es 0. Esto es bastante significativo, puesto que calcular el resto de una división es una operación mucho menos costosa computacionalmente que realizar una división entera por completo.
Por ejemplo, supongamos que tenemos un número natural escrito en base decimal:

$$n = (a_c, a_{c-1}, ..., a_1, a_0)_{10} = \sum_{i=0}^{c} a_i 10^i$$

Entonces, si queremos comprobar si 2 divide a n no necesitamos conocer la división entera de n entre 2, tan solo su resto. Pero como $10, 100, 1000...$ son todos múltiplos de 2, podemos ignorarlos, el resto de dividir n entre 2 es el mismo que el resto de dividir a_0 entre 2. A este tipo de estrategias para determinar si un número es múltiplo de otro se les denomina "criterios de divisibilidad". En particular, el que acabamos de describir es el criterio de divisibilidad por 2.

Definición. Decimos que un número natural n es primo si no es 1 y tiene como únicos divisores a 1 y a n.
Los números naturales que no son primos (ni 1) se llaman números compuestos. Esto se debe a que si un número tiene más de dos divisores, entonces lo podemos expresar como producto de dos números naturales que no son ni 1 ni el propio número.

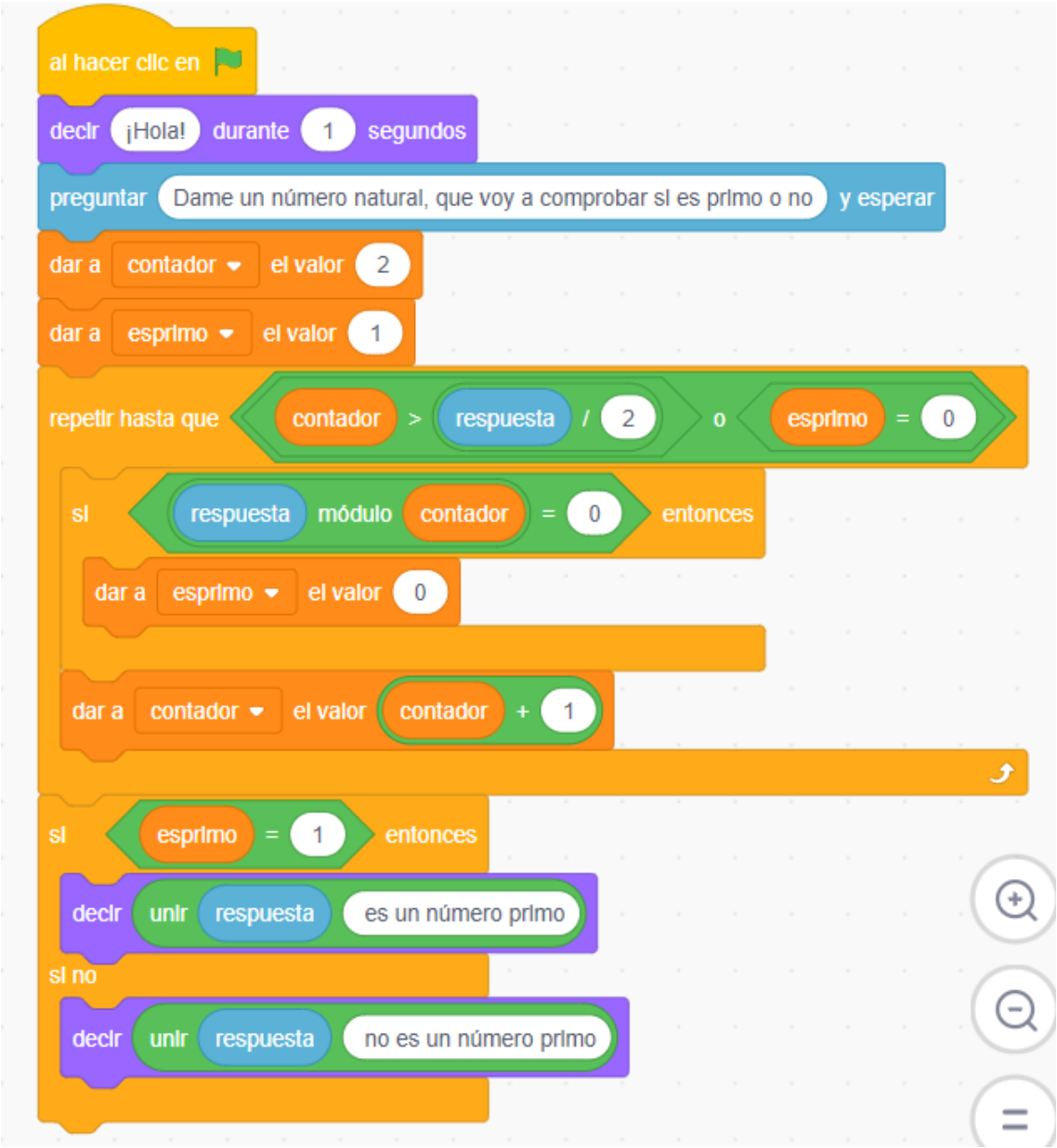
De esta manera, los números primos son los "bloques" primarios con los que construimos los números naturales, utilizando el producto. Los números compuestos son los que tienen más de una "pieza" al construirles de esta manera. Esta idea se expresa de manera más precisa en el siguiente teorema:

Teorema. Todo número natural n mayor que 1 se puede expresar (de manera esencialmente única) como producto de números primos. A esta expresión se le conoce como descomposición de n en factores primos.

Proposición. Si queremos determinar cuáles son los números primos menores o iguales que un cierto natural n, podemos proceder mediante un algoritmo que se conoce como la criba de Eratóstenes. Empezamos escribiendo una lista con los números desde el 2 hasta el n. Después, empezando por 2, tachamos todos los múltiplos (propios) de 2. Sabemos que estos no pueden ser números primos. Continuamos en nuestra lista hasta que lleguemos a un número que no esté tachado (en este primer paso será el 3), y tachamos de la lista todos sus múltiplos propios. Repetimos este paso hasta que lleguemos al número n. Los números que sobreviven en la lista sin ser tachados son exactamente los números primos menores o iguales que n.

3.3.2. El proyecto de Scratch

Ya mostramos anteriormente un programa de Scratch que comprueba si un número natural es primo o no, que se encuentra compartido en `https://scratch.mit.edu/projects/699474244`. Nuestro objetivo sería que los alumnos acabaran desarrollando un programa en Scratch que funcione de la misma manera.
Ahora vamos a proceder a analizar su funcionamiento, así como su relación con los contenidos que vamos a tratar:



Lo primero que hace el programa es pedirnos el número que vamos a comprobar si es primo o no por la pantalla. Una vez introducimos el número por el teclado, comienza el algoritmo propiamente dicho.

En el código contamos con una constante ("respuesta") que es un número entero, la respuesta que introducimos por el teclado, que corresponde al número que estamos comprobando, y a partir de ahora llamaremos n, para abreviar.

Tenemos dos variables de tipo numérico. Una de ellas, "esprimo" la utilizamos como una comprobación de que no hayamos encontrado ningún divisor de n. Por defecto toma el valor 1, como indicación de que al principio del algoritmo aún no tenemos

ningún divisor propio de n, por lo que, con la información de la que disponemos, n podría ser primo. La segunda ("contador"), que llamaremos i a partir de ahora, es una variable que utilizamos como un indicador de que número estamos comprobando si divide a n en cada momento. La variable se inicializa con el valor 2, que es el menor número que debemos comprobar.
El programa tiene dos componentes principales que hacen que la comprobación sea correcta: un bucle "mientras", y un condicional a la salida de este bucle:

1. En el bucle nuestra condición de entrada es que se cumplan dos condiciones: que no hayamos agotado todos nuestros candidatos a divisores (es decir, que sea $i \leq n/2$), y que no hayamos encontrado ningún divisor real de n (o sea, que esprimo valga 1).
   Si se cumplen ambas condiciones, entramos en el bucle, y hacemos lo siguiente: primero comprobamos si i es un divisor de n, y en caso de que lo sea cambiamos el valor de esprimo a 0 (pues ya tenemos información suficiente para deducir que n no es primo). Después de eso, aumentamos en 1 el valor de i.

2. Una vez hemos salido del bucle tenemos un condicional, que, en función del valor de la variable esprimo es capaz de detectar el motivo por el cual hemos salido del bucle:

   - Si esprimo = 1, significa que hemos agotado todos los candidatos a divisores de n, y ninguno de ellos lo era. Por lo tanto, n es un número primo.
   - Si, por lo contrario, esprimo = 0, significa que a lo largo del camino hemos encontrado un número entero (que no es 1 ni n) que divide a n, por lo que n no es un número primo.

Este programa es bastante básico, en el sentido de que no está muy optimizado, hace más comprobaciones de las que son estrictamente necesarias para poder asegurar que un número es primo. En el proyecto `https://scratch.mit.edu/projects/706658868/` hemos recogido dos posibles optimizaciones que le podemos hacer a nuestro código para que tarde menos en comprobar si un número es primo o no:

1. La primera optimización, y la más sencilla de explicar a nuestros alumnos, es que estamos comprobando el doble de los números necesarios, en muchos casos. Esto es debido a que si nuestro número no es múltiplo de 2, tampoco lo será de 4, ni de $6, 8, ...$ Podemos solventar esto inicializando nuestra variable i con el valor 3, y aumentando su valor de 2 en 2 en cada iteración. Para que el programa funcione correctamente de esta manera, debemos hacer la comprobación de si n es par antes de entrar al bucle.

2. La segunda se basa en el hecho de que los divisores de un número "vienen" por parejas. Si m es un divisor de n, entonces $\frac{n}{m}$ es un número entero que también es divisor de n. Esto nos proporciona una biyección entre los divisores de n que son menores o iguales que $\sqrt{n}$ y los que son mayores o iguales que $\sqrt{n}$. Por lo tanto, si hemos comprobado que n no tiene ningún divisor (propio) menor o igual que $\sqrt{n}$, no tiene ninguno en general.

Para comprobar la eficiencia de estas modificaciones, en el proyecto anteriormente mencionado, hago correr el programa que comprueba la primalidad de un número natural en un bucle, comprobando los números naturales de uno en uno, y parando cuando hayan pasado 10 segundos. Los resultados dejan clara la diferencia en eficiencia:

- El programa original comprueba hasta el número 8167.
- Cuando dejamos de buscar divisores pares llegamos hasta el 11987.
- Vemos una enorme diferencia cuando solo comprobamos divisores hasta $\sqrt{n}$, y llegamos hasta el 58985.
- Por último, si juntamos las dos modificaciones nuestro programa comprueba hasta el número 95629.

Mencionar que estos resultados no son siempre iguales, cada vez que ejecutamos el programa durante 10 segundos obtenemos un resultado distinto, pero dentro de un cierto margen de error, que nos permite ver que la diferencia entre la eficiencia de los cuatro algoritmos es notable.

Por último, en relación a esta actividad, hemos implementado en Scratch un algoritmo que realiza la criba de Eratóstenes, utilizando para ello las listas de elementos. El programa, si bien no es excesivamente complicado, es algo largo, así que no incluimos una imagen del código, pero se puede consultar en la página del proyecto, que hemos compartido en la comunidad: `https://scratch.mit.edu/projects/706665816/`.
Si bien las modificaciones a nuestro algoritmo para aumentar su eficiencia solo requieren hacer pequeños cambios en el código original, el algoritmo de la criba de Eratóstenes es un algoritmo completamente distinto, y realizar este programa requeriría por parte de los alumnos algo más de soltura en la programación. A pesar de ello, puede ser una actividad adecuada en las circunstancias correctas, como por ejemplo, como un proyecto a desarrollar en casa, o una actividad de atención a la diversidad, para alumnos de altas capacidades.

3.3.3. Planteamiento

Hemos pensado esta actividad para una clase de matemáticas de segundo de la ESO. Temporalmente, cuándo más sentido tiene es tras finalizar la unidad didáctica correspondiente a los contenidos sobre divisibilidad.
Antes de proceder con la actividad como tal, los alumnos deberían aprender los conceptos básicos de programación en Scratch: los bloques, variables, y los bucles y condicionales, con ejemplos prácticos, pues en caso contrario no serían capaces de desarrollar el algoritmo por su propia cuenta. Esta introducción se realizaría de manera individual en una sesión en un aula de informática.
Después de esto prevemos una duración de dos sesiones más para que los alumnos programen los distintos algoritmos propuestos, trabajando por parejas.

3.3.4. Metodología

Las metodologías que hemos decidido utilizar en esta actividad son:

- Flipped Classroom, o Aula Invertida. En esta metodología, se le da la vuelta al funcionamiento usual de la docencia, es el alumnado quien debe trabajar ciertos conceptos fuera de clase, y esta se dedica a realizar actividades más

prácticas, para potenciar los conocimientos. Se pretende usar esta metodología para la primera sesión de clase, en la que los alumnos estarán aprendiendo el manejo de Scratch. Para ello, el profesor proporcionará a los alumnos un vídeo explicativo sobre la programación en Scratch, que deberán ver por su propia cuenta el día anterior a la sesión de clase correspondiente. Después, en dicha sesión, los alumnos pueden hacer una primera toma de contacto real con el lenguaje, teniendo ya una base teórica y disponiendo del profesor como una guía en este proceso.

- Trabajo por parejas. Hemos pensado que durante las sesiones dedicadas a la elaboración de los distintos programas de Scratch, los alumnos pueden organizarse en parejas elegidas por el docente. De esta manera, ambos alumnos pueden trabajar conjuntamente para buscar la solución, ayudándose mutuamente. Creemos que esta metodología es adecuada, puesto que el profesor puede elegir las parejas de modo que los dos miembros se complementen de la mejor manera posible, permitiendo así que siempre dispongan de varias formas de comprender un algoritmo, o de entender un bloque de programación de Scratch.

3.3.5. Temporalización

La actividad está pensada para tener una duración de tres sesiones de 50 minutos, situadas en un aula de informática, o alternativamente disponiendo de al menos un teléfono o tablet para cada alumno, para que tengan todos ellos acceso a Scratch.

1. En la primera de estas sesiones se corresponde con el aula invertida que hemos descrito en la sección anterior. Se presupone que los alumnos han hecho un cierto trabajo personal en su casa, viendo los vídeos explicativos proporcionados por el profesor sobre la programación en Scratch. El objetivo de esta sesión es realizar tareas que trabajen:

    - Las variables en Scratch. Los estudiantes deben comprender el funcionamiento que tienen las variables en Scratch: cómo definirlas, inicializarlas

con un valor, cambiar este valor, variables numéricas o cadenas de caracteres, etc. Como Scratch permite ejecutar sentencias separadamente, y que el valor de las variables se muestre en tiempo real por la pantalla, esto es sencillo de hacer, los alumnos pueden simplemente probar lo que hacen todos los bloques relativos al bloque de las variables y comprobar el efecto que tienen sobre estas.

- Los bloques relativos a operaciones. El orden de operaciones es importante en Scratch, cada vez que queramos hacer una operación compleja hay que concatenar los bloques de manera correcta. Además, es importante conocer los operadores booleanos. Una buena práctica para conseguir este objetivo es tratar de traducir un polinomio a un bloque en Scratch. Como se presupone que los alumnos ya manejan las variables, pueden darle un valor a una variable, tratar de operar con ella de acuerdo a un polinomio, y comprobar si el valor que obtienen es el mismo que con lápiz y papel.
- Los bloques condicionales. Una vez comprendidos los bloques correspondientes a operaciones booleanas, hay que trabajar sobre su aplicación más importante: los bloques condicionales. Se puede estudiar estos conceptos tratando de elaborar un programa que compruebe si un año es bisiesto o no, o escribiendo un algoritmo que tome como entrada el nombre de un mes y de como respuesta el número de días de ese mes (de esta manera, el alumno debe encadenar varios bloques condicionales, o varios bloques "o" para crear una condición booleana adecuada).
- Los bucles. En Scratch no tenemos bloques específicos correspondientes a los tres tipos de bucles más conocidos (bucle "para", bucle "mientras" y bloque "repetir"), pero se pueden programar de manera sencilla a partir de bloques existentes. Para realizar el programa de los números primos, es suficiente con conocer el bucle "mientras", por lo que centramos nuestra atención en este esquema. Una primera aplicación es la implementación de un bucle "para" con este esquema, es decir, repetir una cierta sentencia un número determinado de veces, utilizando una variable como contador de ese número de veces. Se podría, por ejemplo, hacer un algoritmo que

mostrará por la pantalla los 50 primeros números naturales. Otra aplicación del bucle "mientras" con la que podrían practicar nuestros alumnos es un algoritmo que lea una entrada por pantalla hasta obtener un cierto tipo de dato. Por ejemplo, se podría pedir al usuario que introduzca un número positivo, y repetir la lectura del dato hasta que el dato proporcionado sea realmente un número positivo.

2. En las dos sesiones siguientes se procederá al trabajo por parejas, en el que los alumnos por fin tendrán que desarrollar el programa que comprueba si un número natural es primo o no. Ya presentamos en la sección sobre el proyecto de Scratch un primer algoritmo básico que consigue este cometido, junto con dos posibles optimizaciones del programa, y una actividad adicional sobre la criba de Eratóstenes. Nuestro objetivo es que todos los alumnos lleguen a realizar el primer algoritmo de una manera u otra , aunque la implementación concreta puede variar, y también sería posible que algún alumno implementara una de las optimizaciones directamente, si se le ocurre la idea. Como cada pareja de alumnos trabajará a un ritmo distinto, los alumnos que terminen el programa antes de que finalicen las dos sesiones pueden continuar con el resto de actividades, primero optimizando el programa original para que sea más eficiente (para esto, el profesor les proporcionaría el código del algoritmo que comprueba primalidad durante 10 segundos), hasta que lleguen a deducir, con pistas del docente, las dos optimizaciones que comentamos anteriormente. Si alguna pareja llegara a completar todos los programas en las dos sesiones, el profesor les explicaría como funcionan las listas en Scratch, y les retaría a escribir un programa que realice la criba de Eratóstenes (suponemos que ese algoritmo se ha trabajado ya en clase).
   Realizando la actividad de esta manera estamos atendiendo a la diversidad, puesto que cada par de alumnos aprenderá a un ritmo distinto, y es importante que todos tengan la oportunidad de desarrollar al máximo sus capacidades.

Como ya hemos comentado, la actividad se presenta de forma natural al finalizar la unidad didáctica correspondiente a la divisibilidad, una vez los alumnos tienen

ciertos conocimientos sobre los números primos, y saben perfectamente como determinar si un número natural dado es primo o no.

3.3.6. Evaluación

La forma de evaluar es análoga a la de las actividades anteriores, estaremos utilizando la observación para medir el nivel de participación y esfuerzo de los estudiantes en la actividad, y de la misma manera valoraremos su desempeño en la tarea. Para conseguir esto último, recogeremos el proyecto de Scratch de cada pareja de alumnos, y valoraremos la creatividad, la limpieza del código (presencia de comentarios con explicaciones, organización de los bloques, etc) y la proporción de algoritmos propuestos que hayan sido capaces de programar.

Capítulo 4

Conclusiones

Para concluir este trabajo nos preguntamos cómo de bueno es Scratch como herramienta didáctica en las matemáticas de la ESO, si buscamos fomentar el razonamiento matemático. A la vista de lo expuesto, creemos que se trata de una herramienta didáctica bastante buena si tenemos ese objetivo en mente. Scratch puede ser muy motivante para los alumnos, y hemos mostrado que tiene amplias posibilidades de ser implementado en un aula de la ESO, donde promete ser un recurso divertido, y que desarrolla en los estudiantes habilidades en programación informática y razonamiento matemático. También queda claro que es una herramienta adecuada para los docentes, puesto que permite tanto la posibilidad de tomar actividades hechas por la comunidad (como las expuestas en este trabajo), como la de que cada docente cree por su cuenta una actividad que se adapte a sus necesidades específicas.

Si bien es obvio que no sugerimos una educación basada únicamente en actividades de Scratch (ninguna metodología es adecuada para ser utilizada de manera exclusiva), hemos demostrado que en todos los cursos de la ESO podemos encontrar un lugar en el que dedicar algunas sesiones a actividades basadas en Scratch, dentro de la programación habitual del curso.

Claramente hay una relación bilateral entre matemáticas e informática: por un lado la informática surge inicialmente como una rama de la computación, algo históricamente muy anterior al primer ordenador; y por otro lado diversas ramas de las

matemáticas se han beneficiado enormemente de la informática, algunas incluso no podrían existir de la manera en que las entendemos sin la existencia de ordenadores capaces de realizar una gran cantidad de cálculos por segundo. Creemos que esta relación no está del todo presente en el actual currículo de la ESO, ni tampoco en la forma en la que se aborda a veces la docencia de las matemáticas, pero podría ser bueno hacerla más explícita. Por un lado, los alumnos estarían mejor preparados en un mundo que cada vez está más informatizado, y por otro podría evitar una parte del tedio que son a veces los cálculos en matemáticas: un alumno que es capaz de programar a un ordenador para que resuelva un tipo de problema claramente tiene (a nuestro juicio) una comprensión profunda de éste, por lo que no es necesario que realice esos mismos cálculos a mano innumerables veces.

En resumen, creemos que Scratch es un recurso didáctico a tener en cuenta en la enseñanza de las matemáticas, y esperamos que poco a poco su uso sea cada vez más extendido.

Bibliografía

Bellos, A. (2020). Perilous problems for puzzle lovers . Experiment Llc.

Apuntes Marea Verde https://www.apuntesmareaverde.org.es/

ORDEN EDU/362/2015, de 4 de mayo, por la que se establece el currículo y se regula la implantación, evaluación y desarrollo de la educación secundaria obligatoria en la Comunidad de Castilla y León (2015). BOCyL núm. 86, de 08/05/2015. https://www.educa.jcyl.es/es/resumenbocyl/orden-edu-362-2015-4-mayo-establece-curriculo-regula-implan.ficheros/549394-BOCYL-D-08052015-4.pdf

Real Decreto 217/2022, de 29 de marzo, por el que se establece la ordenación y las enseñanzas mínimas de la Educación Secundaria Obligatoria (2022). BOE núm. 76, de 30/03/2022. https://www.boe.es/eli/es/rd/2022/03/29/217/con

Calao, L.A., Moreno-León, J., Correa, H.E., Robles, G. (2015). Developing Mathematical Thinking with Scratch. In: Conole, G., Klobučar, T., Rensing, C., Konert, J., Lavoué, E. (eds) Design for Teaching and Learning in a Networked World. EC-TEL 2015. Lecture Notes in Computer Science(), vol 9307. Springer, Cham. https://doi.org/10.1007/978-3-319-24258-3_2

Calder, N. y Taylor, M., 2010. Scratching below the surface: Mathematics through an alternative digital lens? Conference: 33rd annual conference of MERGA. At: Freemantle

Kobsiripat, W. (2015). Effects of the Media to Promote the Scratch Programming Capabilities Creativity of Elementary School Students. Procedia - Social And

Behavioral Sciences, 174, 227-232. https://doi.org/10.1016/j.sbspro.2015.01.651

Lopez, V., y Hernandez, M. (2015). Scratch as a computational modelling tool for teaching physics. Physics Education, 50(3), 310-316. https://doi.org/10.1088/0031-9120/50/3/310

López-Escribano, C. y Sánchez-Montoya, R., 2015. Scratch y Necesidades Educativas Especiales: Programación para todos. Revista de Educación a Distancia, Número 34.

Nikou, S., y Economides, A. (2014). Transition in student motivation during a scratch and an app inventor course. 2014 IEEE Global Engineering Education Conference (EDUCON). https://doi.org/10.1109/educon.2014.6826234

Scratch - About. Scratch.mit.edu. (2022). Retrieved 7 June 2022, from https://scratch.mit.edu/about.

Scratch Wiki. En.scratch-wiki.info. (2022). https://en.scratch-wiki.info/.

Smullyan, R. (2008). ¿Cómo se llama este libro?. Cátedra.

Zhang, L. y Nouri, J., 2019. A systematic review of learning computational thinking through Scratch in K-9. Computers & Education, 141, p.103607.