ACTIVIDADES PARA SCRATCH

1.- Básicos

1.1.- Hacer que el gato se sitúe en la esquina inferior izquierda de la pantalla apuntando en dirección horizontal y empiece a moverse realizando rectángulos de 400 píxeles de ancho por 300 de alto hasta que se pulse la tecla “p”. Al terminar cada rectángulo el gato dirá “miau” durante un segundo en una burbuja de texto (no sonido pues en los ordenadores del aula no hay altavoces). Se debe hacer con el lápiz bajado para visualizar la trayectoria.

1.2.- Hacer lo mismo pero el gato debe moverse describiendo triángulos equiláteros de 300 píxeles de lado.

1.3.- Hacer que un objeto empiece a moverse en una dirección aleatoria en línea recta, rebotando en los bordes al chocar.

2.- Interacción

2.1.- Hacer que dos pelotas diferentes empiecen a moverse partiendo de esquinas opuestas de la pantalla en direcciones diferentes elegidas al azar por el programa y rebotando en los bordes. Si chocan se pararán y dirán algún mensaje (por ejemplo una ¡Crash! y otra ¡Ay!) Entonces aparecerá el gato, que habrá estado oculto, y dirá que las pelotas han chocado.

3. Variables

3.1- Modificar el ejercicio 1.1 de modo que una variable contabilice el número de rectángulos que describe el gato y al finalizar éste lo diga en una burbuja de texto.

Además debes conseguir que no haga falta mantener pulsada la tecla “p” hasta que el gato inicie un nuevo rectángulo, sino que una vez pulsada brevemente, se guarde este hecho en alguna variable para que el gato se pare cuando termine el rectángulo que esté haciendo en ese momento.

3.2.- Modifica el ejercicio 1.3 de modo que en vez de moverse siempre en línea recta hasta chocar con bordes, cambie de dirección un ángulo al azar entre 10º y 20º hacia un lado o hacia otro cada 20 pasos.

3.3.- Un programa que resuelva ecuaciones de 2º grado del tipo $ax^2 + bx + c = 0$. Si no tiene solución debe decirlo, si tiene una solución única también y si tiene dos soluciones debe dar ambas.

3.4.- Un programa que haga aparecer dos personajes en pantalla, quietos. Habrá dos pelotas moviéndose por la pantalla, una verde (la de tenis) y una naranja (la de baloncesto). Cuando la pelota verde choca con un personaje, le añade una vida, mientras que cuando la que choca es la naranja, le quita una vida. Cada personaje empieza con 5 vidas. Cada vez que un personaje recibe un pelotazo dice ¡Ay! El primero que se quede sin vidas pierde. El personaje que acaba con sus vidas, dice el mensaje ¡Adiós mundo cruel! y se volcará

Para que no se eternice, podemos hacer la pelota naranja más grande que la verde, para que sea más fácil su choque. Cuando las pelotas choquen con los personajes conviene que reboten (girando, por ejemplo, un ángulo al azar entre 135º y 225º).

3.5.- Varios objetos (bombas por ejemplo) se mueven al azar por la pantalla. Cada vez que uno choca con el gato, le resta una vida de las 7 iniciales (debe emitir un ¡Boom! La bomba y un ¡Ay! el gato). El gato consigue un punto cada vez que consigue chocar con una estrella que habrá en la pantalla. La estrella puede estar fija o moverse al azar rebotando en las paredes. Cada vez que el gato consigue atraparla, aparece en otro punto al azar en la pantalla. El gato se moverá con las teclas de flechas. Las vidas restantes y los puntos obtenidos aparecerán en pantalla.

3.6.- Diseñar un juego en el que el gato empieza en la esquina inferior izquierda de la pantalla y debe avanzar mediante las teclas de flechas hasta alcanzar un premio en la esquina inferior derecha. Cuando la alcanza aparece un nuevo premio en la esquina contraria y tendrá que volver a alcanzarlo, y así irá sumando puntos. Sin embargo, en el camino habrá unas pelotas que se mueven verticalmente rebotando, (pueden ir a diferentes velocidades). Cuando una pelota toca al gato le quita una de las vidas iniciales y cambia momentáneamente de disfraz la pelota. Cuando el gato se queda sin vidas acaba el juego y gana el jugador que haya conseguido más puntos.

b) Complicar el juego de forma que cuando el gato lleve 10 puntos las pelotas vayan más deprisa. O bien cuando haya pasado un cierto tiempo.

4. Diseño de programas más complejos

4.1.- Modificar el programa de ejemplo de los apuntes que pregunta operaciones matemáticas de modo que puedan competir dos jugadores. En pantalla se mostrarán dos personajes inicialmente en el lado izquierdo de la pantalla. El programa debe ir haciendo preguntas sucesivas a cada jugador. Con cada pregunta correcta, su personaje avanza, por ejemplo, 40 pasos hacia la derecha, si falla, retrocederá 20 pasos hacia la izquierda, y se responde la palabra “paso”, se quedará donde está. El primero que llegue a la meta en el lateral derecho ganará. El programa lo anunciará o mostrará al personaje de mayor tamaño o lo que sea.

4.2.- Realizar un programa que simule un cajero automático. Un mensaje informará de que hay que hacer clic sobre un botón para sacar dinero. Al hacer clic sobre dicho botón nos preguntará cuánto dinero queremos sacar. Como sólo tiene billetes de 50, 20 y de 10 €, si le introducimos una cifra que no sea múltiplo de 10 nos dirá que no puede expender dicha cantidad y nos volverá a preguntar. Una vez que le metamos un número válido, el programa debe calcular cuantos billetes de cada tipo nos tiene que dar y nos lo mostrará en pantalla. Tras esto podremos cancelar la operación pulsando sobre un botón Cancelar. Para validar la operación, nos pedirá que pulsemos un botón Validar y, tras ello, volverá a informar de lo que hay que hacer para sacar dinero.

Notas:

a) Tendremos en cuenta que dará preferentemente billetes lo más grandes posibles. Es decir, para darnos 50 € nos lo dará en un único billete de 50 en vez de dos de 20 y uno de 10.

b) Los botones serán objetos sacados de la biblioteca de objetos de Scratch.

c) Conviene informar de las cantidades (dinero pedido, nº de billetes que hay que dar, etc) mostrando las variables en pantalla, mejor que con comentarios de ningún objeto.

4.3.- Vamos a perfeccionar el programa anterior de forma que tenga en cuenta además de las anteriores las siguientes condiciones:

- El cajero inicialmente tendrá un número de billetes de cada tipo en su interior (por ejemplo 10 de cada tipo), de forma que cada vez que entregue billetes los restará de su depósito.
- El cajero seguirá ofreciendo sacar dinero, como en el ejercicio anterior, hasta que no le queden billetes. En este caso, al hacer clic sobre el botón de sacar dinero informará a los usuarios de que no le queda dinero y está fuera de servicio.
- El cajero dispondrá de unas listas con datos del nombre de los clientes, su usuario, sus contraseñas y los saldos que cada uno tiene en sus cuentas (para simplificar podemos hacerlo para 10 clientes, por ejemplo). Cuando se pulsa para sacar dinero pedirá el usuario y, a continuación la contraseña. Si son correctas, saludará con el nombre del cliente y preguntará cuánto dinero desea sacar.

- Si el usuario pide una cantidad de dinero superior a lo que tiene en su cuenta, el cajero le informará de ello, le dirá su saldo y volverá a preguntarle cuánto desea sacar.
- Si el usuario pide una cantidad superior al dinero que le queda al cajero, dirá lo máximo que puede darle y volverá a preguntarle.
- Si el cajero calcula que con los tipos de billetes disponibles no puede conseguir la cantidad deseada, informará de ello y solicitará una nueva cantidad.
- Una vez que la operación sea válida (está verificado el cliente, tiene saldo suficiente, el cajero tiene dinero suficiente, etc,) la operación podrá cancelarse haciendo clic sobre un botón Cancelar (de esta forma no restará el dinero del saldo del cliente ni del cajero) o validarse haciendo clic sobre un botón Validar (el dinero sacado por el cliente se restará de su saldo y al dinero disponible en el cajero y a los tipos de billetes que hayan salido).

4.4.- Ejercicio sobre **puerta de garaje**.

Se te da un archivo scratch donde ya hay una serie de objetos. Se trata de simular el funcionamiento de una puerta de garaje de apertura automática.

- Inicialmente la puerta estará cerrada, el piloto rojo encendido y el verde apagado.
- Al hacer clic sobre el botón Abrir, la puerta irá desplazándose hacia la derecha hasta que el garaje quede abierto. Entonces, al final de recorrido, se apagará el piloto rojo y se encenderá el verde. La puerta no debe desaparecer del todo, sino quedar visto un filo.
- Cuando se haga clic sobre el botón Cerrar, la puerta se desplazará en sentido contrario. El piloto verde se apagará desde el inicio del recorrido y el rojo se encenderá. Así hasta que la puerta quede cerrada.
- Tanto en un caso como en otro la velocidad de la puerta debe ser uniforme.

Otras variantes del problema que lo van complicando que se proponen:

1.- Que además de hacer clic sobre los botones que también pueda activarlos el gato desplazándose y tocando uno u otro.

2.- En caso de que la roca se desplace y se ponga en la trayectoria de la puerta, cuando está cerrando, la puerta al tocar la roca debe reaccionar abriendo de nuevo y quedarse abierta viéndose un filito, como antes (hay que tener en cuenta que ahora el recorrido de la puerta es diferente).

3.- Si estando cerrando la puerta se vuelve a activar el botón Abrir, la puerta debe empezar a abrir de nuevo, hasta quedarse abierta como antes (viéndose un filito). La puerta debe ir siempre a la misma velocidad.

4.- Si estando abriendo la puerta se activa el botón Cerrar, la puerta volverá a cerrarse antes de haber terminado el recorrido de apertura. La puerta debe ir siempre a la misma velocidad

4.5.- Ejercicio sobre parking. Se entrega archivo de scratch con los objetos ya incluidos.

Disponemos de un parking de pago con seis plazas. Tiene una barrera de entrada y una de salida. Cada barrera está horizontal cuando está cerrada y abre girando 90º en torno a su base, colocándose vertical.

Junto a cada barrera hay un semáforo de dos luces (roja y verde). Estará encendida la roja y apagada la verde con la barrera cerrada o en proceso de cierre o apertura, y estará encendida la verde y apagada la roja cuando la barrera esté totalmente abierta. Cada luz puede tener, por tanto, dos disfraces, uno con un color más oscuro y otro con el mismo color más brillante.

Junto a cada barrera hay un botón (triángulo amarillo) para indicar el deseo de entrar o salir del parking.

Disponemos de 8 coches que, al empezar el programa, están en la calle, fuera del parking, y a un tamaño reducido. Cuando los coches pasan al parking deben cambiar a un tamaño un poco mayor, y al salir de nuevo a la calle, reducirán de nuevo su tamaño.

Cuando un coche se selecciona haciendo clic sobre él cambia de disfraz, apareciendo un recuadro azul alrededor de él. Al seleccionar otro coche, el anteriormente seleccionado, si lo hubiera, se deselecciona.

El parking tendrá un coste que dependerá del tiempo que ha estado dentro cada coche. Será el resultado de pagar el nº de segundos que haya estado por el precio del segundo que lo almacenaremos en una variable, para poder cambiarlo fácilmente cuando queramos

Funcionamiento:

Cuando queramos introducir uno de los coches que están en la calle dentro del parking lo seleccionaremos y a continuación haremos clic sobre el triángulo amarillo de la entrada.

Si no quedaran plazas libres, no se abrirá la barrera y aparecerá un mensaje intermitente durante un breve tiempo indicándolo. Si hay plazas libres el coche aumentará un poco de tamaño y aparecerá ante la barrera de entrada, ésta se levantará, el semáforo verde se encenderá y, a continuación, el vehículo se situará en alguna plaza libre. El programa debe buscar las plazas libres y dirigir automáticamente el coche hacia ella. Puede simplificarse el programa haciendo aparecer el coche directamente en la plaza libre sin realizar el recorrido.

Cuando queramos que un coche salga del parking, lo seleccionaremos y haremos clic sobre el botón triangular amarillo de la salida. El coche se dirigirá automáticamente hasta la salida (por simplificar puede hacerse aparecer el coche directamente en la salida sin realizar el recorrido), se indicará durante unos segundos lo que debe pagar, se levantará la barrera, se encenderá el semáforo verde y el coche saldrá, dirigiéndose otra vez al recuadro que simula a la calle.

Hay que tener cuidado de cuando seleccionemos un coche dentro del parking, el programa no haga nada si se hace clic sobre el botón de entrada. E, igualmente, si se selecciona un coche que está en la calle, el programa no haga nada si se hace clic sobre el botón de salida.

Sugerencias para resolver el problema:

- Se puede crear variable tipo lista para identificar a cada coche. Como si fuesen sus matrículas. Las matrículas pueden ser inicialmente los colores de los coches, para que sea más fácil durante la realización del programa. Así, si el coche morado es el primero de la lista matrícula, nos referiremos a él en el programa como elemento 1 de matrícula. Esto tiene la ventaja de que una vez realizados los programas para un coche, se pueden copiar fácilmente a los otros coches sin más que cambiar el nº del elemento.
- Conviene otra lista donde se indique, para cada coche que entra en el parking, su hora de entrada, consultando el cronómetro en el momento de entrar. Así, volviendo a consulta el cronómetro en el momento de salir, y restando podemos saber cuántos segundos ha estado.
- También convendría crear dos listas para guardar las posiciones x e y respectivamente de los coches en la calle, para situarlos al empezar el programa siempre en el mismo sitio y cada vez que salgan del parking.
- También puede ser útil crear una lista para indicar si cada coche está en la calle o en el parking. Así, cuando lo seleccionemos podemos saber donde está el coche. Así, por ejemplo,

si seleccionamos un coche que ya está dentro del parking, el programa sólo ejecutará órdenes de salida pero ignorará órdenes de entrada. Y a la inversa si el coche está en la calle.

- Otra variable tipo lista que puede ser interesante es una que indique para cada plaza si ésta está libre u ocupada. Por ejemplo, se puede introducir el valor 0 cuando está libre y la matrícula del coche que la ocupa cuando esté ocupada. Esto nos facilitará que el programa pueda localizar una plaza libre buscando una que tenga un 0 en esta lista.
- Si vamos a hacer el programa en su forma más fácil, es decir, omitiendo el recorrido de los vehículos tanto para entrar como para salir, nos puede interesar una lista que nos indique la posición x de cada plaza (ya que la posición y es la misma para todas en este caso) para así indicar dónde tiene que desplazarse el coche que entra.
- Una variable necesaria seguro que es la que guarde el coche que está seleccionado en el momento en que se pulsa el botón de entrada o de salida, para saber el coche que hay que mover o del que calcular el importe a pagar.