**Creando Objeto Aleatorio**

Como objetivo vamos a realizar un juego de tiro al blanco con movimientos aleatorios, con la finalidad de clickear en el centro y generar una alerta.



Para generar el sorteo de nuestro tiro al blanco debemos generar numero aleatorios con la función sorteo posición(maximo) al recibir un parámetro de limite, los números generados estarán dentro del rango de nuestra pizarra.

Cada vez que creemos nuestro circulo será con las coordenadas aleatorias.

**Disparando contra el objetivo**

Nuestro objetivo ahora es crear una función que permita obtener la posición del área central del circulo con la finalidad de que cuando hagamos click poder obtener los mismos resultados.

Para realizar la comparación entre donde hice el click y la posición del tiro al blanco debemos recuperar los datos de nuestro click con la función disparar. En esta función recopilamos las posiciones de x e y de nuestro plano para luego comprar si x o y son iguales a la posición aleatoria de nuestro programa.

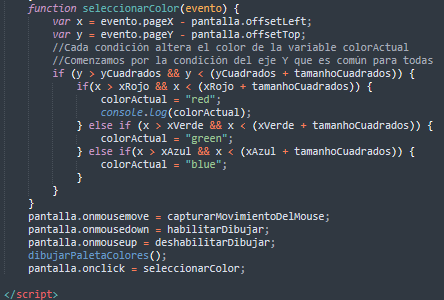
Recordemos que:

* **Math.round:** la usamos cuando queremos retornar el valor de un número redondeado al entero más cercano.
* **Math.floor:** cuando queremos redondear un número al máximo entero menor.

**Paint**

El siguiente código permite dibujar círculos en la pantalla mientras el botón izquierdo se encuentra presionado, eso significa que mientras no soltemos el botón izquierdo, diseñaremos un círculo a lado de otro, que dará un efecto como si estuviéramos pasando un pincel por la pantalla. Si soltamos el botón izquierdo, la acción de mover el mouse sobre la pantalla no tendría que diseñar nada. Al final tendremos un efecto visto en herramientas como Paint de Windows.

El código es el siguiente:



Para desarrollar este programa reutilizamos todo el código que ya teníamos para dibujar circunferencias, y solo adicionamos la lógica nueva de transformar nuestro mouse en un pincel.

Considera que la parte esencial está en saber si diseñamos o no en la pantalla mientras pasamos el mouse sobre la misma. Y sabemos que la condición está asociada a si el botón izquierdo del mouse está o no siendo presionado.

Siendo así declaramos una variable booleana llamada puedoDibujar que comienza como false. Esa variable será utilizada por la función dibujarCirculo para saber si debe o no diseñar.

En el código habrás visto tres eventos nuevos que estamos usando, ellos son onmousemove, onmousedown y onmouseup, donde el primero permite capturar el movimiento del mouse, el segundo sirve para ejecutar un código cuando el mouse está presionado y el tercero cuando el botón del mouse es soltado. Entonces, asociamos el primer evento a la función para dibujar circulo: pantalla.onmousemove = dibujarCirculo. Además, creamos dos funciones habilitarDibujar y deshabilitarDibujar y asociamos respectivamente cada función con los eventos del mouse onmousedown y onmouseup. Dentro de las funciones, lo que hacemos es atribuir la variable puedoDibujar para true cuando queremos habilitarDibujar y vuelve para false cuando queremos deshabilitarDibujar.

Lo primero que resolveremos, será crear las variables y la función que usaremos para crear un cuadrado genérico.

En esta primera parte, definimos las coordenadas de nuestros cuadrados var xRojo = 0, var xVerde = 50, var xAzul = 100, todos ellos tienen la misma coordenada vertical var yCuadrados = 0, y por último, el tamaño de los cuadrados de 50 píxeles var tamanhoCuadrados = 50. Luego, creamos la función genérica dibujarCuadrado() que recibe como parámetro las coordenadas (x,y), el tamaño y el color del cuadrado.

En una segunda etapa vamos a crear una función que dibuje la paleta de colores, que se encargue de llamar a la función dibujarCuadrado y le pase los parámetros correctos para graficar los 3 cuadrados, nuestro código quedaría así:

En la tercera parte de nuestro problema, nos vamos a enfocar en la lógica para que al momento de hacer clic el color del pincel cambie, esta última parte, la vamos a subdividir en dos partes, la primera nos enfocaremos en no permitir que el usuario diseñe encima de nuestra paleta de colores, y una segunda parte en detectar cual cuadrado fue accionado con el clic del mouse.

Para resolver que el usuario no esté permitido de diseñar encima de nuestra paleta, aplicaremos la siguiente lógica, si el mouse se encuentra posicionado entre las coordenadas X=0 y X=3\*(tamanhoCuadrados+5), y entre Y=0 y Y=(tamanhoCuadrados+5), no podrá diseñar, incluso cuando el botón izquierdo del mouse esté presionado. Toma en cuenta que sumamos el radio de la circunferencia también, porque en caso de que el usuario grafique en el borde inferior o borde derecho de la paleta corremos el riesgo de que se grafique una parte del círculo dentro de la paleta. Para aplicar la lógica mencionada, vamos a crear una función llamada puedeDisenharArea con todas las condiciones, esta función retornará true caso sea posible y false caso contrario. La función recibirá como parámetro las coordenadas X y Y del mouse.

Pero para poder accesar a las coordenadas del mouse necesitamos usar el evento onmousemove que es parte de nuestro objeto pantalla, dicho evento anteriormente estaba asociado directamente a la función dibujarCirculo, ahora lo vamos a asociar con una nueva función que crearemos: capturarMovimientoDelMouse y que recibirá como parámetro el evento de mover el mouse.

En esta función capturarMovimientoDelMouse vamos a guardar las variables de las coordenadas X y Y, y dentro de la misma función vamos a crear un if para verificar si el mouse está localizado en el área restringida, si no estuviera vamos a llamar a la función dibujarCirculo.

Necesitamos, también, alterar la función dibujarCirculo, ya que ahora recibirá como parámetro las coordenadas (x,y) del clic del mouse cuando deseemos graficar.

En la parte dos vamos a seleccionar el color, Para ello, lo primero que haremos será crear una variable colorActual y le atribuimos el color blue por defecto, luego vamos a detectar cuál color de la paleta fue seleccionado a través de la función seleccionarColor que asociaremos al evento onclick de nuestra pantalla. Voy a explicar la lógica para el color rojo, para los otros cuadrados la lógica es la misma. Si el usuario hace clic en la coordenada X=15 y Y=20, necesitamos verificar si X es mayor que xRojo, que es la posición del eje X del cuadrado rojo, y además necesitamos verificar si X es menor que xRojo + tamanhoCuadrados porque ese es el range del cuadrado rojo en el eje horizontal. Pero también necesitamos verificar el eje vertical, para ello, vamos a comparar si Y es mayor que yCuadrados y que Y sea menor que yCuadrados + tamanhoCuadrados. Ojo, también necesitamos cambiar la función dibujarCirculo porque tuvimos que adicionar la variable del color seleccionado colorActual entre los parámetros recibidos, hicimos esa alteración en la declaración de la función dibujarCirculo y en la llamada a la función dibujarCirculo que está localizada dentro de la función capturarMovimientoDelMouse.

Lo que aprendimos:

* A crear un juego interactivo.
* A redondear números con una nueva función Math.floor().
* A abstraer un problema para proponer una solución lógica a través de un algoritmo.
* A usar varios conceptos de lógica en un solo programa, como funciones, condiciones, capturar eventos, ciclos *Loop*, etc.