CÍRCULO REBOTANDO III

Processing

Lenguaje de programación basado en Java y C++ pensado para diseño digital e iniciación a la programación.

IDE

Son las iniciales de Entorno de Desarrollo Integrado, una aplicación informática para facilitarle al programador el desarrollo de software. Normalmente consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. Algunos IDE contienen un compilador y/o un intérprete.

24) Cambiamos el tamaño

Partidos del programa de la práctica II, que puedes descargar desde este enlace.

Ahora queremos que los círculos cambien de tamaño cuando chocan entre ellos. Para ello debemos crear 3 variables tipo entero (int) que serán las que contengan el valor del tamaño de cada uno de los círculos:

```
7
8
9  //tamaño
10 int tam1=80, tam2=40, tam3=60; //tamaño
11
12 //color
13 color coloruno=#F53939;
14 color colordos=#34EA75;
15 color colortres=#2D89F7;
```

También crearemos otras 3 variables con el radio de cada círculo, ya que después nos serán de utilidad:

```
8
9  //tamaño
10 int tam1=80, tam2=40, tam3=60; //tamaño
11 int rad1,rad2,rad3;
12
13 //color
14 color coloruno=#F53939;
15 color colordos=#34EA75;
```

Y sustituimos el valor numérico del tamaño por la variable que acabamos de crear:

```
43
44 // circulo1
45 fill(coloruno); //color de relleno blanco
46 ellipse(posX,posY,tam1,tam1); //dibuja circulo
47 posY=posY+velY;
48 posX=posX+velX;
49
```

Lo mismo para los 3 círculos. Ejecutamos el programa y comprobaremos que funciona exactamente igual que antes, simplemente cambiamos la manera de indicar el tamaño de los círculos.

25) Definimos los radios

Establecemos el valor de las variables de los radios:

```
41
42 void draw() {
43   background(255); //color de fondo
44
45   rad1=tam1/2; //radio circulo 1
46   rad2=tam2/2;
47   rad3=tam3/2;
48
49   // circulo1
50   fill(coloruno); //color de relleno
```

Lo haremos dentro de la función draw, para que al estar dentro del bucle principal del programa se actualice en función del tamaño de los círculos.

26) Cambio de tamaño en el choque

Ahora añadiremos el código necesario para que cuando dos de los círculos reboten cambien su tamaño, de manera que el que sea más grande disminuya su tamaño y viceversa:

```
77   // rebote entre los circulos
78   if(dist(posX,posY,posX2,posY2)<=60) {
79         tam1-=4;
80         tam2+=4;
81         velY=-velY;
82         velX=-velX;
83         velY2=-velY2;
84         velX2=-velX2;
85         auxiliar=coloruno;
86         coloruno=colordos;
87         colordos=auxiliar;
88       }
```

La instrucción tam1-=4; es lo mismo que poner tam1=tam1-4;

Simplemente resta 4 pixeles al tamaño del círculo, siendo 80-76-72-....cada vez que choque con otro círculo.

Lo mismo con tam2+=4; es lo mismo que tam2=tam2+4;

incrementos/decrecimientos

Cuando queremos que aumente una variable podemos poner: **a=a+1;** o también **a++;** o incluso **a+=1;** . Son tres modos de poner los mismo.

Del mismo modo podemos actuar con los decrecimientos:

a=a-1; **a- -;** **a-=1;**

27) Rebote entre los círculos

Si ejecutamos el programa veremos como los círculos al rebotar cambian su tamaño, pero si dejamos que lo hagan varias veces veremos que llega un momento en que rebotan antes de que se toquen. Esto es debido a que al cambiar el tamaño también cambia el tamaño entre sus centros:

```
76
77    // rebote entre los circulos
78    if(dist(posX,posY,posX2,posY2)<=60) {
79          tam1-=4;
80          tam2+=4;
81          velY=-velY;
```

En el caso anterior, cuando el círculouno tenía tamaño 80, o sea, radio 40 px, y el círculos tamaño 40, radio 20, la distancia entre sus centros sería la suma de los radios 40+20=60, por eso era el valor de referencia. Debemos sustituir ese valor fijo por la fórmula rad1+rad2 , para que lo actualice en función de la ejecución del programa:

```
76
77    // rebote entre los circulos
78    if(dist(posX,posY,posX2,posY2)<=rad1+rad2) {
79          tam1-=4;
80          tam2+=4;
81          velY=-velY;
82          velX=-velX;
```

Y lo mismo para los otros dos posibles choques...

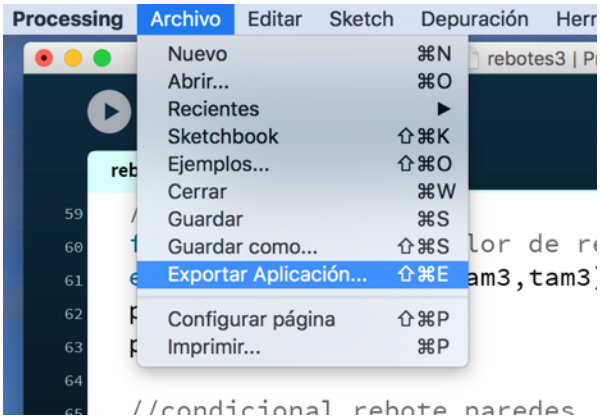
```
61  ellipse(posX3,posY3,tam3,tam3); //dibuja circulo
61    ellipse(posX3,posY3,tam3,tam3); //dibuja circulo
62    posY3=posY3+velY3;
63    posX3=posX3+velX3;
64
65    //condicional rebote paredes
66    if(posY>360-rad1|| posY<rad1) {velY=-velY;}
67    if(posX>640-rad1 || posX<rad1) {velX=-velX;}
68
69    if(posY2>360-rad2 || posY2<rad2) {velY2=-velY2;}
70    if(posX2>640-rad2 || posX2<rad2) {velX2=-velX2;}
71
72    if(posY3>360-rad3 || posY3<rad3) {velY3=-velY3;}
73    if(posX3>640-rad3 || posX3<rad3) {velX3=-velX3;}
74
75    // rebote entre los circulos
76    if(dist(posX,posY,posX2,posY2)<=rad1+rad2) {
77          tam1-=4;
```

28) Y también los rebotes contra las paredes

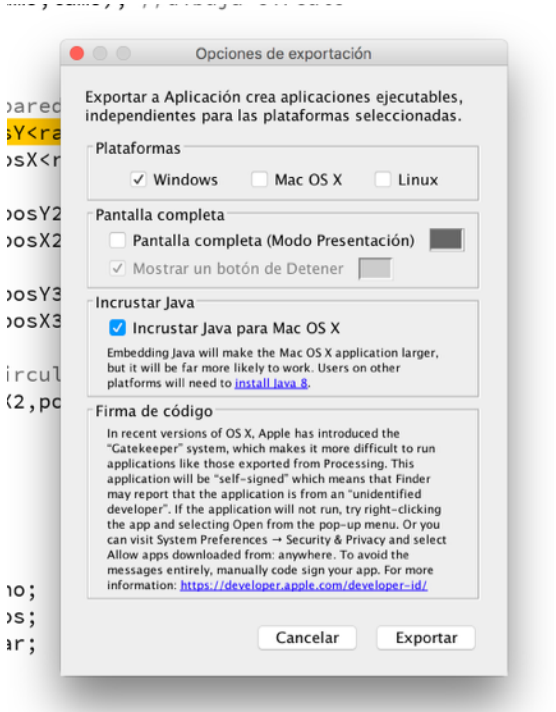
Y ahora hacemos lo mismo con los condicionales donde los círculos rebotan contra los laterales de la ventana del programa, tal como podemos ver en la imagen de la izquierda.

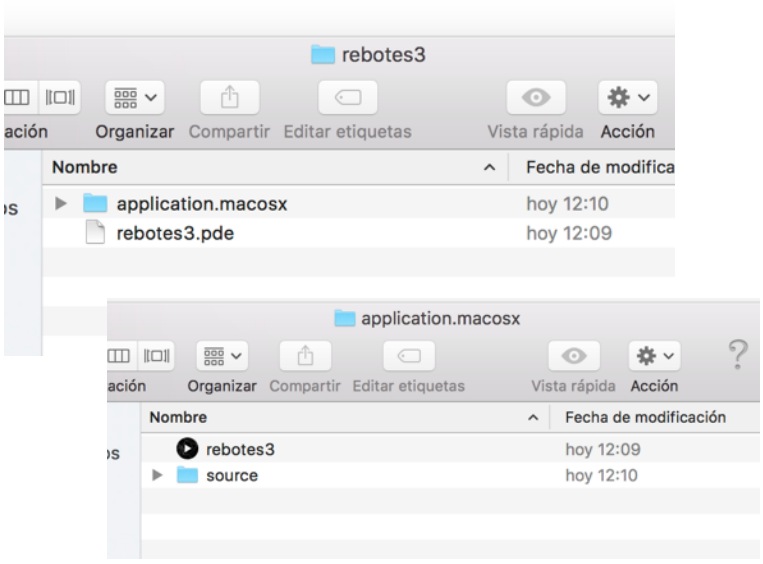
28) Y creamos nuestra aplicación

Una vez que comprobamos que todo funciona correctamente ya podemos crear nuestra aplicación, totalmente ejecutable y multiplataforma (windows, linux o mac). Para ello debemos ir al menú de *Archivo* y seleccionar *Exportar Aplicación*:



Elegiremos el sistema operativo o sistemas operativos para los que queremos hacer la aplicación, si queremos que se ejecute a pantalla completa y si queremos que incluya los archivos necesarios de Java para que funcione:





Si exportamos el archivo nos creará una carpeta *application.sistema operativo*, en la que incluirá el código fuente y el archivo ejecutable, en este caso **rebotes3**

Con esto finalizamos esta práctica...