

CÍRCULO REBOTANDO II

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 facilitar 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.

15) Añadimos un nuevo círculo

Partidos del programa de la práctica 1, que podemos descargar: [Pulsa aquí](#).

Tenemos un círculo que rebota contra los bordes de nuestra ventana. Lo que vamos a hacer ahora es crear otro círculo que haga lo mismo que el anterior. Para ello debemos duplicar las variables y todas las ordenes de movimiento, condicionales del primer círculo:



```

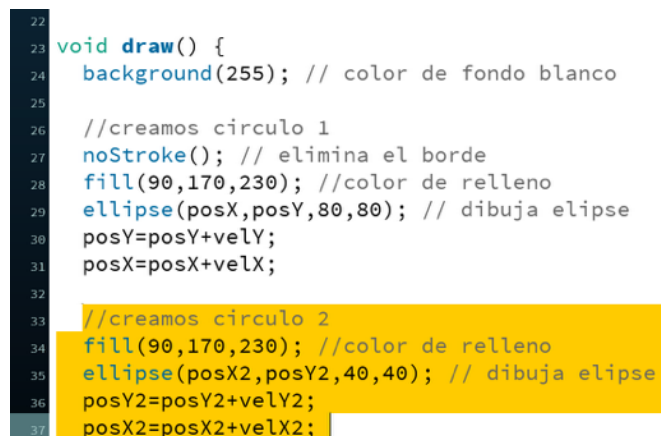
2 * @author jorge gomez
3 * @version 1.0
4 */
5
6 // variables circulo 1
7 int posX=320;
8 int posY=180;
9 int velX=1; //velocidad eje x
10 int velY=1; //velocidad eje y
11
12 // variables circulo 2
13 int posX2=80; //posicion inicial en x
14 int posY2=320; // posicion inicial en y
15 int velX2=-1; //velocidad eje x
16 int velY2=1; //velocidad eje y
17
18 void setup(){
19   size(640,360); // tamaño ventana
20

```

Les añadimos un 2 en el nombre para diferenciarlas, y les damos diferentes valores iniciales.

16) Nuevo círculo

Establecemos el tamaño del nuevo círculo en 40 px:



```

22
23 void draw() {
24   background(255); // color de fondo blanco
25
26   //creamos circulo 1
27   noStroke(); // elimina el borde
28   fill(90,170,230); //color de relleno
29   ellipse(posX,posY,80,80); // dibuja elipse
30   posY=posY+velY;
31   posX=posX+velX;
32
33   //creamos circulo 2
34   fill(90,170,230); //color de relleno
35   ellipse(posX2,posY2,40,40); // dibuja elipse
36   posY2=posY2+velY2;
37   posX2=posX2+velX2;
38

```

Límites rebote

Como el tamaño de nuestro segundo círculo es de 40 px, por lo tanto 20 px de radio, para hacer el efecto de rebote contra el borde debemos restar el radio de los límites:

$$360 - 20 = 340$$

$$640 - 20 = 620$$

Y sumarlo en el otro extremo, por eso ponemos 20.



dist
en esta orden escribiremos entre paréntesis cuatro parámetros, la posición inicial X e Y y la final X e Y:

dist(Xi,Yi,Xf,Yf);

17) Y los rebotes

Duplicamos los condicionales para el rebote del segundo círculo contra los bordes de nuestra ventana. Recordemos que nuestra ventana tiene un tamaño de 640x 360 pixeles:

```

38
39
40 //rebote contra los laterales
41 if(posY>320 || posY<40){velY=-velY;}
42 if(posX>600 || posX<40){velX=-velX;}
43 if(posY2>340 || posY2<20){velY2=-velY2;}
44 if(posX2>620 || posX2<20){velX2=-velX2;}
45

```

18) Rebote entre los círculos

Ahora vamos a hacer que nuestros círculos reboten entre ellos. Para ello tenemos que tener en cuenta el radio de cada uno, 40 y 20 px respectivamente. Cuando chocan significa que la distancia que les separa es igual o menor a 60 (la suma de los radios), que mediremos con la orden dist:

```

39
40 //rebote contra los laterales
41 if(posY>320 || posY<40){velY=-velY;}
42 if(posX>600 || posX<40){velX=-velX;}
43 if(posY2>340 || posY2<20){velY2=-velY2;}
44 if(posX2>620 || posX2<20){velX2=-velX2;}
45
46 // rebote entre los dos círculos
47 if(dist(posX,posY,posX2,posY2)<=60){
48     velY=-velY;
49     velX=-velX;
50     velY2=-velY2;
51     velX2=-velX2;
52 }
53
54 }

```

Si ejecutamos nuestro programa veremos como los círculos rebotan contra las paredes y también lo hacen entre ellos.

19) Y añadimos otro círculo

Y ahora hacemos lo mismo con otro círculo de tamaño 60, por lo tanto radio 30 px, repitiendo los pasos anteriores... por lo que tendremos tres círculos rebotando entre ellos y con los límites de la ventana.

```

10 int velY=1; //velocidad eje y
11
12 // variables circulo 2
13 int posX2=80; //posicion inicial en x
14 int posY2=320; // posicion inicial en y
15 int velX2=-1; //velocidad eje x
16 int velY2=1; //velocidad eje y
17
18 // variables circulo 3
19 int posX3=600; //posicion inicial en x
20 int posY3=60; // posicion inicial en y
21 int velX3=1; //velocidad eje x
22 int velY3=-1; //velocidad eje y

```

variables

creamos círculo

```

38
39 //creamos circulo 2
40 fill(90,170,230); //color de relleno
41 ellipse(posX2,posY2,40,40); // dibuja elipse
42 posY2=posY2+velY2;
43 posX2=posX2+velX2;
44
45 //creamos circulo 3
46 fill(90,170,230); //color de relleno
47 ellipse(posX3,posY3,60,60); // dibuja elipse
48 posY3=posY3+velY3;
49 posX3=posX3+velX3;
50
51 //rebote contra los laterales
52 if(posY>320 || posY<40){velY=-velY;}

```

```

55 if(posY2>340 || posY2<20){velY2=-velY2;}
56 if(posX2>620 || posX2<20){velX2=-velX2;}
57
58 if(posY3>330 || posY3<30){velY3=-velY3;}
59 if(posX3>610 || posX3<30){velX3=-velX3;}
60

```

rebote paredes

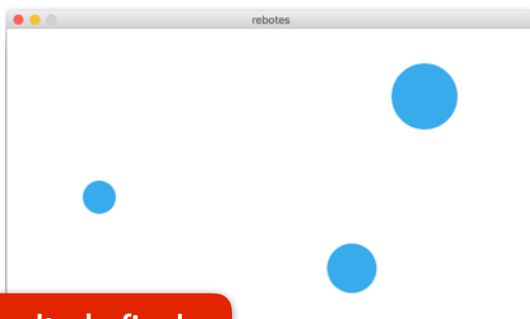
rebote entre círculos

```

61 // rebote entre los circulos 1 y 2
62 if(dist(posX,posY,posX2,posY2)<=60) {
63     velY=-velY;
64     velX=-velX;
65     velY2=-velY2;
66     velX2=-velX2;
67 }
68 // rebote entre los circulos 2 y 3
69 if(dist(posX3,posY3,posX2,posY2)<=50) {
70     velY3=-velY3;
71     velX3=-velX3;
72     velY2=-velY2;
73     velX2=-velX2;
74 }
75 // rebote entre los circulos 1 y 3
76 if(dist(posX,posY,posX3,posY3)<=70) {
77     velY=-velY;
78     velX=-velX;
79     velY3=-velY3;
80     velX3=-velX3;
81 }

```

resultado final



random

Podemos usar la instrucción `random` con un solo parámetro o con dos:

```
random(100)
```

Generará un valor aleatorio entre 0 y 100

```
random(10,40)
```

Generará un valor aleatorio entre 10 y 40

20) Inicio aleatorio

Vamos a utilizar la instrucción `random` que generará un número aleatorio entre los límites que nosotros le indiquemos. Simplemente tenemos que hacer previamente un cambio, la orden `random` creará números tipo `float` (decimales), por lo que debemos cambiarlo al declarar la variable:

```
14
15 //variables
16 float posX=random(40,600);
17 float posY=random(40,320);
18 int velX=1; //velocidad en eje X
19 int velY=1; //velocidad en eje Y
20
21 float posX2=random(20,620);
22 float posY2=random(20,340);
23 int velX2=-1; //velocidad en eje X
24 int velY2=1; //velocidad en eje Y
25
26 float posX3=random(30,610);
27 float posY3=random(30,330);
28 int velX3=1; //velocidad en eje X
29 int velY3=-1; //velocidad en eje Y
```

Al ejecutar el programa podemos comprobar como aparecen los círculos en diferentes lugares de la ventana.

21) Color para los círculos

Ahora haremos que los círculos sean de diferentes colores, y que cuando choquen entre ellos se los intercambien. Para ello usaremos un tipo de dato primitivo que es `color`, que nos sirve para almacenar el color en código hexadecimal o RGB.

```
7
8
9 //color
10 color coloruno=#F53939; // rojo
11 color colordos=#34EA75; // verde
12 color colortres=#2D89F7; //azul
13 color auxiliar; // sin determinar
14
15 //variables
16 float posX=random(40,600);
```

Creamos una variable para cada círculo y una más llamada *auxiliar* que la usaremos para el intercambio de color.

Ahora sustituimos el valor del color de relleno (fill) por las variables creadas:

```

36 void draw() {
37   background(255); //color de fondo
38
39   // circulo1
40   fill(coloruno);
41   ellipse(posX,posY,80,80); //dibuja circulo
42   posY=posY+velY;
43   posX=posX+velX;
44
45   // circulo2
46   fill(colordos);
47   ellipse(posX2,posY2,40,40); //dibuja circulo
48   posY2=posY2+velY2;
49   posX2=posX2+velX2;
50
51   // circulo3
52   fill(colortres);
53   ellipse(posX3,posY3,60,60); //dibuja circulo
54   posY3=posY3+velY3;
55   posX3=posX3+velX3;

```

22) Cambio de color

Ahora añadiremos en donde tenemos definido el choque de los círculos las instrucciones necesarias para que se intercambien el color:

Intercambio de colores

Auxiliar pasa a ser **coloruno**, coloruno pasa a ser **colordos**, y por último colordos pasa a ser auxiliar, que era coloruno

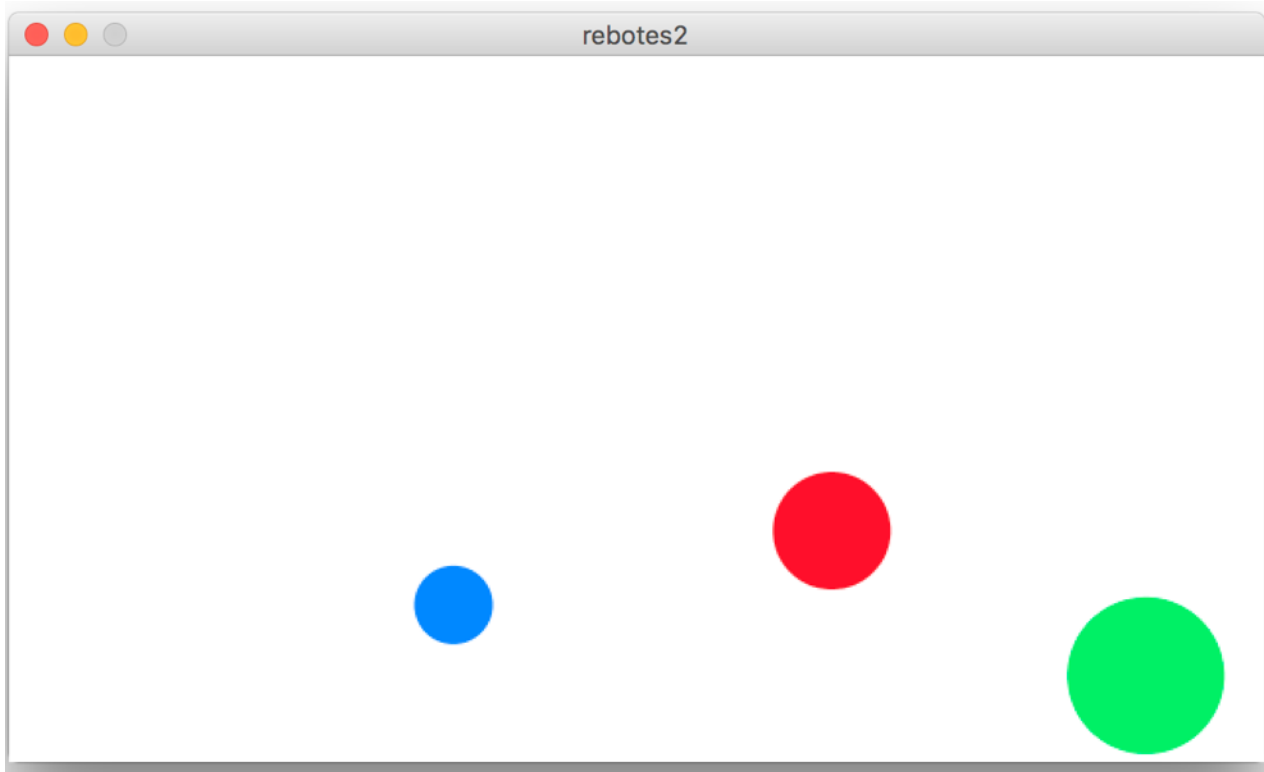
```

67 // rebote entre los circulos
68 if(dist(posX,posY,posX2,posY2)<=60) {
69   velY=-velY;
70   velX=-velX;
71   velY2=-velY2;
72   velX2=-velX2;
73   auxiliar=coloruno;
74   coloruno=colordos;
75   colordos=auxiliar;
76 }
77
78
79 if(dist(posX,posY,posX3,posY3)<=70) {
80   velY=-velY;
81   velX=-velX;
82   velY3=-velY3;
83   velX3=-velX3;
84   auxiliar=coloruno;
85   coloruno=colortres;
86   colortres=auxiliar;
87 }
88
89 if(dist(posX2,posY2,posX3,posY3)<=50) {
90   velY2=-velY2;
91   velX2=-velX2;
92   velY3=-velY3;
93   velX3=-velX3;
94   auxiliar=colordos;
95   colordos=colortres;
96   colortres=auxiliar;
97 }

```

23) *Final*

Este es el final de la práctica, donde tenemos tres círculos de colores, que chocan entre ellos y que al hacerlo intercambian sus colores.



Puedes descargar el programa de esta práctica desde [este enlace](#).