Tema 16: ANIMACIONES

Vamos a comenzar con el movimiento más simple: una bola representada por un círculo rojo, que inicialmente está en la parte superior de la pantalla y se mueve hacia abajo a una velocidad constante. Cuando llegue al borde inferior invertiremos su velocidad, y se volverá hacia arriba. Al llegar arriba, volveremos a invertir su velocidad y el movimiento se repetirá.
Se dibujará sobre el canvas directamente.

BolaBotadoraUniforme:BOLA BOTADORA MOVIMIENTO UNIFORME:

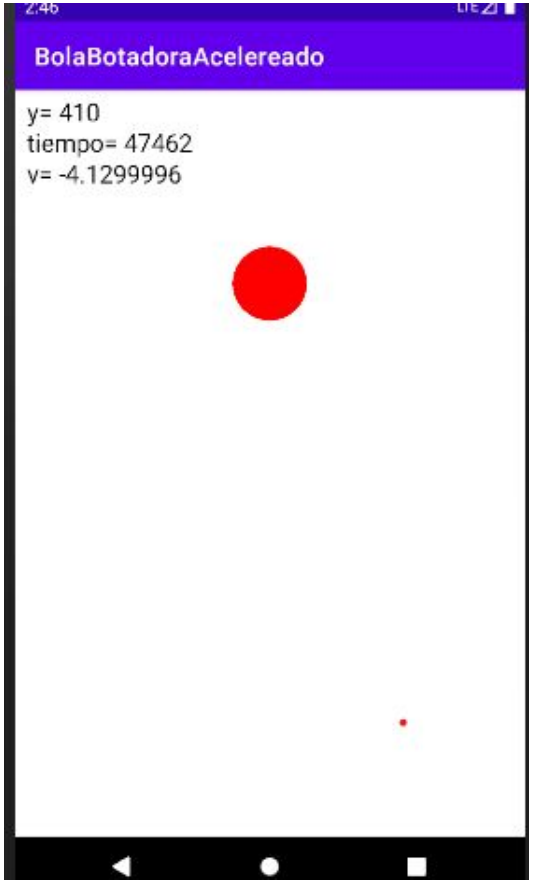
```
public class MainActivity extends AppCompatActivity {
boolean continuar=true;
float velocidad=1.5f;
int dt=10;
int tiempo=0;
Thread hilo=null;
DinamicaView dinamica;
float s;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        dinamica=new DinamicaView(this);
        setContentView(dinamica);
        s=getResources().getDisplayMetrics().density;
        hilo=new Thread(dinamica);
        hilo.start();
    }
    //detenemos el hilo si pausa
    @Override
    public void onPause(){
        super.onPause();
        continuar=false;
    }
    //reiniciamos el hilo si resume
    @Override
    public void onResume(){
        super.onResume();
        continuar=true;
        if(!hilo.isAlive()){
            hilo=new Thread(dinamica);
            hilo.start();
        }
    }
```

```
class DinamicaView extends View implements Runnable{
    int x,y,ymax;
    Paint paintFondo,paintParticula,paint;
        public DinamicaView(Context context) {
            super(context);
            //Colores para el dibujo y el tamaño del texto
            paintFondo=new Paint();
            paintParticula=new Paint();
            paint=new Paint();
            paintFondo.setColor(Color.WHITE);
            paintParticula.setColor(Color.RED);
            paint.setColor(Color.BLACK);
        }
        @Override
        public void run() {
            while(continuar){
                tiempo=tiempo+dt;
                //movimiento rectilineo uniforme y=y+v*t
                y=y+(int)(velocidad*dt);
                //si llega abajo invertimos la velocidad:
                if(y>ymax){
                    velocidad=-velocidad;
                }
                //si llega arriba invertimos la velocidad:
                if(y<0){
                    velocidad=-velocidad;
                }
                postInvalidate();
                try{Thread.sleep(dt);}
                catch (InterruptedException e){;}
            }
        }
```

```
//obtiene geometria del canvas
        @Override
        protected void onSizeChanged(int
w,int h,int oldw,int oldh){
            x=w/2;
            y=0;
            ymax=h;
        }
        @Override
        public void onDraw(Canvas canvas){
            canvas.drawPaint(paintFondo);
            paint.setTextSize(20*s);

canvas.drawCircle(x,y,30*s,paintParticula);
            canvas.drawText("y=
"+y,10*s,25*s,paint);
            canvas.drawText("tiempo=
"+tiempo,10*s,50*s,paint);
        }
    }//fin clase DinamicaView
}
```

BOLA BOTADORA MOVIMIENTO ACELERADO



BolaBotadoraAcelerado: Tenemos que introducir en el código una aceleración:

```
float velocidad=1.5f;
float aceleracion=0.01f;
int dt=10;

//Se introduce el cambio de velocidad en cada intervalo de tiempo según la fórmula v=v_o+a*dt
velocidad=velocidad+aceleracion*dt;
```

JUEGO BOLA BOTADORA REBOTADORA: modificando la anterior aplicación, conseguir que la bola botadora rebotadora se mueva en cualquier dirección. Ahora se moverá en ambos ejes: x e y, no sólo hacia arriba y abajo.
VER: https://www.flipandroid.com/bola-de-rebote-de-android.html

INSERTAR IMÁGENES EN UN CANVAS: es posible insertar imágenes dibujandolas en un canvas, dentro del método onDraw de una clase View. Tendremos control total sobre la posición y el tamaño y se pueden manipulaciones gráficas. Para dibujar una imagen podemos usar un objeto de la clase Drawable, referenciando el fichero de recursos con la imagen: (habitualmente en el constructor de la clase View):

```
Drawable imagen =ContextCompat.getDrawable(getApplicationContext(), R.drawable.mifichero);
```

La imagen se dibujará en el método onDraw, definiendo antes sus dimensiones:

```
imagen.setBounds(x1,y1,x2,y2);
imagen.draw(canvas);
```

Los puntos (x1,y1) y (x2,y2) son los vértices opuestos del rectángulo que contendrá la imagen en la pantalla. Si esos puntos no se asignan, la imagen no se verá en la pantalla, pues por defecto son cero.
Para obtener las dimensiones de un objeto Drawable se usan los métodos:

```
int ancho= ContextCompat.getDrawable(context,R.drawable.mifichero).getIntrinsicWidth();
int alto=ContextCompat.getDrawable(context,R.drawable.mifichero).getIntrinsicHeight();
imagen.setBounds(0,0,ancho,alto);
```

Las dimensiones intrínsecas del drawable no corresponden a las dimensiones reales de la imagen en píxeles. Al construir el drawable, la imagen se escalará internamente; su anchura y altura se multiplicarán por la densidad del dispositivo (`float d=getResources().getDisplayMetrics(). density;`). De esta forma, la imagen parecerá aproximadamente del mismo tamaño en distintos dispositivos.

KILLBUG: Realizar una aplicación “Aplasta Bichos”: aparecerán 5 imágenes en movimiento y el jugador irá aplastando cada imagen. Tras aplastarla al imagen cambiará (bicho aplastado o similar) y desaparecerá tras unos segundos. Añadir sonido.
ADIVINA número: aparece un número deslizándose desde arriba y hay que pulsar el botón correspondiente si es par o impar.
TAREA: Aplicación 2D con movimiento.