Hoạt hình với hình vẽ

Tiếp theo bài trước.

Bài này sẽ làm hoạt hình cho đối tượng bitmap. Ta chạm vào màn hình và xuất hiện hình, hình đó sẽ duy chuyển quanh màn hình, hướng di chuyển và tốc độ là ngẫu nhiên. Hoạt hình dựa vào khái niệm khung hình mỗi giây (FPS). FPS sẽ hiện thị ở góc trên bên trái màn hình để ta theo dõi.

1. Trong lớp "element.java" đầu tiên thêm vào 2 biến mSpeedX, mSpeedY để điều khiển tốc độ theo 2 truc X, Y.

```
    private int mSpeedX;
    private int mSpeedY;
```

2. Trong hàm tạo, tạo ra đối tượng Random để lấy random cho 2 biến mSpeedX, mSpeedY.

```
    public element (Resources res,int x,int y)
    {
           Random rand=new Random();

           mybitmap=BitmapFactory.decodeResource(res,R.drawable.ic_launcher);
           mX= x-mybitmap.getWidth()/2;
           mY= y-mybitmap.getHeight()/2;

           mSpeedX=rand.nextInt(7)-3;
           mSpeedY=rand.nextInt(7)-3;
    }
```

3. Xây dựng một hàm mới tên animate() . Hàm này sẽ thay đổi vị trí của element dựa vào tốc độ và thời gian trôi qua kể từ lần gọi trước đó. (hàm checkborder ta sẽ viết ở bước tiếp theo).

```
    public void animate(long thoigian)
    {
           mX+=mSpeedX * (thoigian/20f);
           mY+=mSpeedY * (thoigian/20f);
           checkborder();
    }
```

4. Khi đối tượng chạm vào các cạnh của màn hình ta sẽ thay đổi hướng chạy của đối tượng (giống như bị đụng và tưng qua hướng khác). Ta sẽ xây dựng 1 hàm tên checkborder() để xét khi đụng 1 trong 4 cạnh. Nhưng trước tiên ta quay qua class Draw2d.java và sửa tên 2 biến mX,mY thành mWidth, mHeight.

```
    public static float mWidth;
    public static float mHeight;
```

5. Tiếp theo ta thêm vào hàm surfaceChanged như sau. Sau khi thêm code thì giờ ta đã có mWidth và mHeight là kích thước chiều rộng và chiều cao của vùng vẽ.

```
public void surfaceChanged(SurfaceHolder holder, int format, int width,
        int height) {
    // TODO Auto-generated method stub
    mWidth=width;
    mHeight=height;

}
```

6. Quay lại class "element.java" giờ ta sẽ viết hàm checkborder() để xét khi chạm 4 cạnh như sau:

```
    private void checkborder()
    {
        if(mX<=0)
        {
            mSpeedX=-mSpeedX;
            mX=0;
        }
        else if(mX+mybitmap.getWidth()>=Draw2d.mWidth)
        {
            mSpeedX=-mSpeedX;
            mX=(int)Draw2d.mWidth-mybitmap.getWidth();
        }
        if(mY<=0)
        {
            mSpeedY=-mSpeedY;
            mY=0;
        }
        if(mY+mybitmap.getWidth()>=Draw2d.mHeight)
        {
            mSpeedY=-mSpeedY;
            mY=(int)Draw2d.mHeight-mybitmap.getHeight();
        }
    }
```

7. Quay về file "Draw2d.java" thêm 1 biến toàn cục chứa số phần tử và một biến thuộc Paint

```
private int mElementNumber = 0;
private Paint mPaint = new Paint();
```

8. Trong "Draw2d.java" Hàm doDraw cũng phải sửa lại. Truyền thêm 1 biến thoigian kiểu long và dùng drawText để xuất thông báo số FPS lên góc trên bên trái.

```
    protected void doDraw(long thoigian,Canvas canvas) {
        // TODO Auto-generated method stub
        super.onDraw(canvas);

        canvas.drawColor(Color.BLACK);

        synchronized(taphop)
```

```
		{
			for(element phantu : taphop)
				phantu.doDraw(canvas);

		}
		canvas.drawText("FPS: " + Math.round(1000f / thoigian) + " Elements: " +
mElementNumber, 10, 10, mPaint);

	}
```

9. Trong hàm onTouchEvent, thêm vào khi một phần tử thêm vào biến taphop thì mElementNumber cũng được cập nhật để lấy tổng số phần tử trong taphop.

```
	@Override
	public boolean onTouchEvent(MotionEvent event) {
		// TODO Auto-generated method stub
		synchronized(taphop)
		{
			element e=new
element(getResources(),(int)event.getX(),(int)event.getY());
			taphop.add(e);
			mElementNumber=taphop.size();
		}
		return super.onTouchEvent(event);
	}
```

10. Ta thấy rằng hàm animate() trong class chưa ai gọi nó. Nó sẽ được gọi trong vòng lặp của thread. Trong file "Draw2d.java" xây dựng 1 hàm tên animate() (Chú ý: hàm animate này là của Draw2d nó sẽ duyệt tất cả phần tử element vào gọi đến hàm animate() của lớp element, tức là có 2 hàm animate() của 2 class khác nhau).

```
	public void animate(long thoigian)
	{
		synchronized(taphop)
		{
			for(element phantu: taphop)
			{
				phantu.animate(thoigian);
			}
		}
	}
```

11. Qua file ViewThread.java khai báo thêm 2 biến

```
	private long thoigianbatdau;
	private long thoigiandaqua;
```

12. Cũng trong ViewThread.java sửa hàm run lại như sau:

```
	@Override
	public void run() {
		// TODO Auto-generated method stub
```

```
        super.run();
        thoigianbatdau=System.currentTimeMillis();
        Canvas canvas=null;
        while(mRun)
        {
                canvas=mHolder.lockCanvas();
                if(canvas!=null)
                {
                        mdraw2d.animate(thoigiandaqua);
                        mdraw2d.doDraw(thoigiandaqua,canvas);
                        thoigiandaqua=System.currentTimeMillis()-thoigianbatdau;
                        mHolder.unlockCanvasAndPost(canvas);
                }
                thoigianbatdau=System.currentTimeMillis();
        }
}
```

13. Chạy chương trình để thấy kết quả, mỗi lần chạm sẽ phát sinh 1 đối tượng, đối tượng chạy theo 1 hướng ngẫu nhiên với thời gian cũng ngẫu nhiên (do đó sẽ thấy chạy giật giật). Và khi chạm vào màn hình sẽ dội lại hướng khác. Góc trên bên trái sẽ hiện số FPS và số phần tử. Nếu không hiện có thể Paint vẽ trùng màu với vùng vẽ. Quay lại hàm Draw2d, trong hàm tạo bổ sung thêm lệnh để set màu cho Paint.

```
        mPaint.setColor(Color.WHITE);
```