TEMA 17: SurfaceView

Para mostrar gráficos propios podríamos usar un componente que herede de `View`. Estos componentes funcionan bien si no necesitamos realizar repintados continuos o mostrar gráficos 3D.

Sin embargo, en el caso de tener una aplicación con una gran carga gráfica, como puede ser un videojuego, un reproductor de vídeo, o una aplicación que muestre gráficos 3D, en lugar de `View` deberemos utilizar `SurfaceView`. Esta última clase nos proporciona una superficie en la que podemos dibujar desde un hilo en segundo plano, lo cual libera al hilo principal de la aplicación de la carga gráfica.

La clase surfaceView proporciona una superficie de dibujo integrada dentro de una jerarquía de vistas. Puedes controlar el formato de esta superficie y, si lo deseas, su tamaño; SurfaceView se encarga de colocarla en la ubicación correcta de la pantalla.

Uno de los propósitos de esta clase es proporcionar dicha superficie en la que un hilo secundario podrá renderizarse en la pantalla. Si vas a utilizarlo de esta manera, debes tener en cuenta algunas semánticas de subprocesos.

Para crear una vista con `SurfaceView` tendremos que crear una nueva subclase de dicha clase (en lugar de `View`). Pero en este caso no bastará con definir el método `onDraw`, ahora deberemos crearnos un hilo independiente y proporcionarle la superficie en la que dibujar (`SurfaceHolder`). Además, en nuestra subclase de `SurfaceView` también implementaremos la interfaz `SurfaceHolder.Callback` que nos permitirá estar al tanto de cuando la superficie se crea, cambia, o se destruye.

VER ejemplo:
http://edu4java.com/es/androidgame/androidgame2.html

El uso de esta clase nos permitirá tener un objeto donde dibujar sobre un hilo secundario; y por otro lado, los métodos de la interface *SurfaceHolder.Callback* nos harán saber cuando la superficie se crea, cambia o se destruye, esto nos ayudará a saber cuando podemos empezar a dibujar, hacer cambios o detenernos.
Cuando la superficie sea creada pondremos en marcha nuestro hilo de dibujado, y lo pararemos cuando la superficie sea destruida. A continuación mostramos un ejemplo de dicha clase:

```
public class VistaSurface extends SurfaceView  implements SurfaceHolder.Callback {
    HiloDibujo hilo = null;

    public VistaSurface(Context context) {
        super(context);

        SurfaceHolder holder = this.getHolder();
        holder.addCallback(this);
    }

    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
        // La superficie ha cambiado (formato o dimensiones)
    }

    public void surfaceCreated(SurfaceHolder holder) {
        hilo = new HiloDibujo(holder, this);
        hilo.start();
    }

    public void surfaceDestroyed(SurfaceHolder holder) {

        hilo.detener();
        try {
            hilo.join();
        } catch (InterruptedException e) { }
    }}
```

Como vemos, la clase `SurfaceView` simplemente se encarga de obtener la superficie y poner en marcha o parar el hilo de dibujado. En este caso la acción estará realmente en el hilo, que es donde especificaremos la forma en la que se debe dibujar el componente. Vamos a ver a continuación cómo podríamos implementar dicho hilo:

```
class HiloDibujo extends Thread {
    SurfaceHolder holder;
    VistaSurface vista;
    boolean continuar = true;

    public HiloDibujo(SurfaceHolder holder, VistaSurface vista) {
        this.holder = holder;
        this.vista = vista;
        continuar = true;
    }

    public void detener() {
        continuar = false;
    }

    @Override
    public void run() {
        while (continuar) {
            Canvas c = null;
            try {
                c = holder.lockCanvas(null);
                synchronized (holder) {
//Actualizar la lógica de juego
                  update();
 // Dibujar aqui los graficos: doDraw(c);
                    c.drawColor(Color.BLUE);
                }
            } finally {
                if (c != null) {
                    holder.unlockCanvasAndPost(c);
                }
            }
        }   }}
```

Podemos ver que en el bucle principal de nuestro hilo obtenemos el lienzo (Canvas) a partir de la superficie (SurfaceHolder) mediante el método lockCanvas. Esto deja el lienzo bloqueado para nuestro uso, por ese motivo es importante asegurarnos de que siempre se desbloquee. Para tal fin hemos puesto unlockCanvasAndPost dentro del bloque finally. Además debemos siempre dibujar de forma sincronizada con el objeto SurfaceHolder, para así evitar problemas de concurrencia en el acceso a su lienzo.

Para aplicaciones como videojuegos 2D sencillos un código como el anterior puede ser suficiente (la clase View sería demasiado lenta para un videojuego). Sin embargo, lo realmente interesante es utilizar SurfaceView junto a OpenGL, para así poder mostrar gráficos 3D, o escalados, rotaciones y otras transformaciones sobre superficies 2D de forma eficiente.

EJEMPLO SurfaceView1: Crear un juego en 2D que permite al usuario mover un círculo rojo tocando la pantalla:

```
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
public class GameSurfaceView extends SurfaceView implements
SurfaceHolder.Callback {
    private GameThread gameThread;
    private Paint paint;
    private float playerX, playerY;
    public GameSurfaceView(Context context, AttributeSet attrs) {
        super(context, attrs);
        getHolder().addCallback(this);
        // Configuración de eventos táctiles y de teclado
        setFocusable(true);
        // Inicialización de objetos y recursos
        paint = new Paint();
        paint.setColor(Color.RED);
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        // Inicialización de hilo de juego
        gameThread = new GameThread(holder);
        gameThread.setRunning(true);
        gameThread.start();
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
        // Ajustes cuando cambia el tamaño
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        // Detener el hilo de juego y liberar recursos
        boolean retry = true;
        gameThread.setRunning(false);
        while (retry) {
            try {
                gameThread.join();
                retry = false;
            } catch (InterruptedException e) {
                e.printStackTrace();
            }       }   }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        // Manejar eventos táctiles
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                playerX = event.getX();
                playerY = event.getY();
                break;
            case MotionEvent.ACTION_MOVE:
                playerX = event.getX();
                playerY = event.getY();
                break;
        }
        return true;
    }
    private class GameThread extends Thread {
        private SurfaceHolder surfaceHolder;
        private boolean running = false;
        public GameThread(SurfaceHolder holder) {
            surfaceHolder = holder;
        }
        public void setRunning(boolean run) {
            running = run;
        }
        @Override
        public void run() {
            while (running) {
                Canvas canvas = surfaceHolder.lockCanvas();
                if (canvas != null) {
                    // Operaciones de dibujo en el lienzo
                    drawGame(canvas);
                    surfaceHolder.unlockCanvasAndPost(canvas);
                }           }       }   }
    private void drawGame(Canvas canvas) {
        // Dibujar en el canvas
        canvas.drawColor(Color.BLACK); // Fondo negro
        canvas.drawCircle(playerX, playerY, 50, paint); // Jugador como un círculo rojo
    }
}
```

EJERCICIOS

SurfaceView1: bola roja sobre fondo negro que se mueva hacia donde se haga click

SurfaceView2: bola roja que se va moviendo por la pantalla. Un cuadrado azul aparece en una posición aleatoria desde la derecha y va hacia la izquierda en línea recta. La bola debe evitar chocar; podrá hacer un salto al pulsar click.

PRÁCTICA DE JUEGO 2D USANDO LA CLASE SurfaceView, Y CAPTURA DE EVENTOS.

Algunas ideas:

- personaje que se mueve a través de un laberinto; hay recompensas y penalizaciones.
- un círculo que se mueve a través de la pantalla en todas las direcciones, hay un cuadrado que aparece por la derecha y desaparece por la izquierda y no se pueden tocar. El círculo dará un salto para evitar al cuadrado tras pulsar un click.
- juego del ping pong