**Drawing an image using SurfaceView**

In this little tutorial we are going to see how to print an image in the android screen directly. First we are going to extend a *View*and then we are going to use a *SurfaceView* object that is a more direct (low level) way. We are going to use *SurfaceView*because we want complete control over the screen.

|  |  |
| --- | --- |
|  |  |

**Create a new android project**

*Project name:* killthemall-training

*Application name:* Kill Them All

*Package name:* com.edu4java.android.killthemall

*Activity:* Main

**Step 1 priting with onDraw using View**

**package** com.edu4java.android.killthemall;

**import** android.app.Activity;

**import** android.os.Bundle;

**import** android.view.Window;

**public** **class** Main **extends** Activity {

    /\*\* Called when the activity is first created. \*/

    @Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

        requestWindowFeature(Window.*FEATURE\_NO\_TITLE*);

        setContentView(**new** GameView(**this**));

    }

}

**package** com.edu4java.android.killthemall;

**import** android.content.Context;

**import** android.graphics.Bitmap;

**import** android.graphics.BitmapFactory;

**import** android.graphics.Canvas;

**import** android.graphics.Color;

**import** android.view.View;

**public** **class** GameView **extends** View {

**private** Bitmap bmp;

**public** GameView(Context context) {

**super**(context);

            bmp = BitmapFactory.*decodeResource*(getResources(), R.drawable.*icon*);

      }

      @Override

**protected** **void** onDraw(Canvas canvas) {

          canvas.drawColor(Color.*BLACK*);

          canvas.drawBitmap(bmp, 10, 10, **null**);

      }

}



In the *View* implementation the *onDraw* method is called directly by the holder in a code we can not see when the *View* is created. The holder is the object that contains the *View*.

When we implement the version extending *SurfaceView* the *onDraw* method is not called. Then we have to call it explicitly.

In order to call *onDraw* we need the *Canvas* object to pass as a parameter. You can think in the *Canvas* as a blackboard where we can draw what we want. Another problem is when everything is ready to start painting the *Canvas*.

The solution is to create a callback listener to the holder. In the *surfaceCreated* method (that is called when the view has been created) we get the *Canvas* from the holder and then call the *onDraw* method.

To get the *Canvas* we use the method *lockCanvas* that locks the *Canvas* in order to avoid anyone else drawing when you are*drawing*. Once you finish drawing we have to unlock the *Canvas* calling *unlockCanvasAndPost* method.

We have to take into account that during the time we have a resource locked we are losing performance and it is important to minimize the locking time.

**Step 1 priting with onDraw using SurfaceView**

**public** **class** GameView **extends** SurfaceView {

**private** Bitmap bmp;

**private** SurfaceHolder holder;

**public** GameView(Context context) {

**super**(context);

             holder = getHolder();

             holder.addCallback(**new** SurfaceHolder.Callback() {

                    @Override

**public** **void** surfaceDestroyed(SurfaceHolder holder) {

                    }

                    @Override

**public** **void** surfaceCreated(SurfaceHolder holder) {

                           Canvas c = holder.lockCanvas(**null**);

                           onDraw(c);

                           holder.unlockCanvasAndPost(c);

                    }

                    @Override

**public** **void** surfaceChanged(SurfaceHolder holder, **int** format,

**int** width, **int** height) {

                    }

             });

             bmp = BitmapFactory.*decodeResource*(getResources(), R.drawable.*icon*);

       }

       @Override

**protected** **void** onDraw(Canvas canvas) {

             canvas.drawColor(Color.*BLACK*);

             canvas.drawBitmap(bmp, 10, 10, **null**);

       }

}