P82 Mobile Application Development - Android

Custom Views

Custom Views

3 main types of custom views:

- From an existing view
- By combining several existing views
- By recreating it from scratch from class View

How views work

- Placement or layout (not to be confused)
- Place the views in the graphical interface (2 steps)

  - Each layout will give size instructions to children

void measure (int widthMeasureSpec, int heightMeasureSpec)

  - Layout gives final dimension to children

void layout (left int, right int, int bottom)

    - A ViewGroup

http://blog.denevell.org/android-custom-views-onlayout-onmeasure.html

How views work

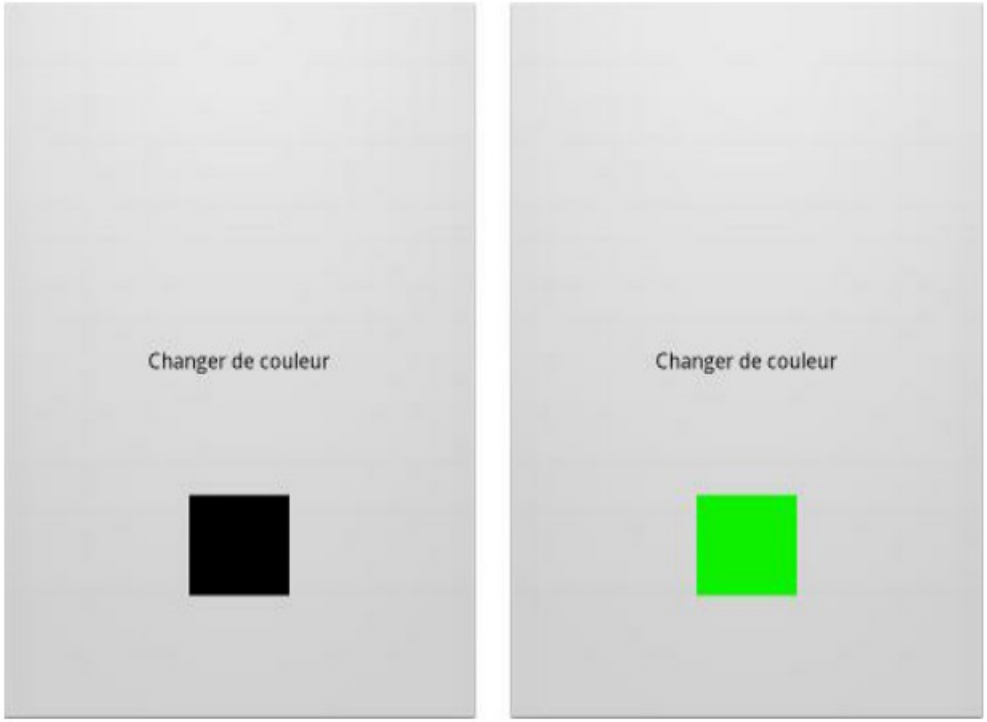
- Drawing

  - Placement and dimension force the view to be drawn using the draw(Canvas canvas) method

- When and how to redraw a view::
  - Views don't draw themselves they require information from:
    - Constructor
    - By programming
  - If we have to force redraw, we call its method
    - Invalidate() -> whole view
    - Invalidate(int left, int top, int right, int bottom) -> partial
    - Only child in invalidate region will be redraw (spread)

Edit an existing view

- We will redefine callback methods
- Example: a button with a colored edge that changes when you click on it



Custom view 1

- Inherits from the Button class

```
public class MySuperButton extends Button {
    //color array
    int[] color;
    //rectangle that we are going to color
        Rect mRect = null;
    Paint paint;
    int position = 0;
    //Event on click
    OnClickListener myListener;
    …
    }
```

Custom view 1

- Define the constructor

```
public MySuperButton(Context context, String txtButton) {
        super(context);

        color = new int[]{Color.RED, Color.BLUE, Color.GREEN};
        setText(txtButton);
        paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setColor(color[0]);

        myListener = new OnClickListener() {
        @Override
        public void onClick(View v) { Log.d("test", "test");
                position++;
                position %= color.length;
                paint.setColor(color[position]);
                invalidate();
                }
        };
        setOnClickListener(myListener);
}
```

Custom view 1

- Redefines onLayout

```
@Override
protected void onLayout(boolean changed, int left, int top, int
right, int bottom) {
    if (changed) {
        // size of the area to draw
        mRect = new Rect(0, 0, getWidth/2, getWidth/2);
    }
    super.onLayout(changed, left, top, right, bottom);
}
```

Custom view 1

- Redefines drawing

```
@Override
   protected void onDraw(Canvas canvas) {
       canvas.drawRect(mRect, paint);
       super.onDraw(canvas);
   }
```

View Composition

- Create a control with a TextView, an EditText and a Button
- Inherit from LinearLayout

```
public class MySuperComposition extends
   LinearLayout {
    private EditText et;
    private TextView tv;
    private Button btn;
    private Context ctx;
   }
```

Custom view 2

- linearLayout Constructor

```
public MySuperComposition(Context context, String myText) {
        super(context);
        ctx = context;

        // viewGroup info
        setOrientation(HORIZONTAL);

        setLayoutParams(new
    ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATC
    H_PARENT, ViewGroup.LayoutParams.WRAP_CONTENT));
        setBackgroundColor(Color.RED);
```

Custom view 2

- TextView and EditText Constructor

```
//TextView
    tv = new TextView(ctx);
    tv.setLayoutParams(new
    LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT ,
    ViewGroup.LayoutParams.WRAP_CONTENT, 30));
    tv.setText(myText);

// EditText
    et = new EditText(ctx);
    et.setLayoutParams(new
    LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT ,
    ViewGroup.LayoutParams.WRAP_CONTENT,50));
    et.setHint("Enter text here");
```

Custom view 2

- Button Constructor and add to view

```
//Button
    btn = new Button(ctx);
    btn.setLayoutParams(new
    LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
    ViewGroup.LayoutParams.WRAP_CONTENT,10));
    btn.setText("OK");
    btn.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            Toast.makeText(ctx,et.getText(),Toast.LENGTH_SHORT).show();
        }
    });

//Add to view
    addView(tv);
    addView(et);
    addView(btn);
```

From a View

- Example: remake a view that displays smaller and smaller squares with choice of colors in the XML file
- Link : http://developer.android.com/training/custom-views/index.html

Custom view 3

- Inherit from View and add two constructors

```
public class MyView extends View {
	private Paint mPaintOne = null;
	private Paint mPaintTwo = null;
	private Context ctx;

	// for Java
	public MyView (Context context, int color1, int color2) {
		super (context);
	}

	// for the XML
	public MyView (Context context, AttributeSet attrs) {
		super (context, attrs);
	}
}
```

Custom view 3

- Declare attributes if you want to use the view in XML

```
<!--same name as the View -->
<declare-styleable name="MyView">
        <!-- The identifier attribute "colorOne" is of type
"color" -->
        <attr name="colorOne" format="color" />
        <attr name="colorTwo" format="color" />
    </declare-styleable>
```

Custom view 3

- Redefining the JAVA constructor

```
public MyView (Context context, int color1, int color2) {
        super (context);
        mPaintOne = new Paint (Paint.ANTI_ALIAS_FLAG);
        mPaintOne.setColor (color1);

        mPaintTwo = new Paint (Paint.ANTI_ALIAS_FLAG);
        mPaintTwo.setColor (color2);

        ctx = context;
}
```

Custom view 3

- Redefining the XML constructor

```
public MyView (Context context, AttributeSet attrs) {
        super (context, attrs);
        TypedArray attr = context.obtainStyledAttributes (
                attrs,
                R.styleable.MyView);

        mPaintOne = new Paint (Paint.ANTI_ALIAS_FLAG);
        mPaintOne.setColor (attr.getInt (R.styleable.MyView_colorOne,
        Color.BLACK));

        mPaintTwo = new Paint (Paint.ANTI_ALIAS_FLAG);
        mPaintTwo.setColor (attr.getInt (R.styleable.MyView_colorTwo,
        Color.GRAY));

        attr.recycle ();
        ctx = context;
}
```

Custom view 3

Redefine onMeasure()

```
@Override
protected void onMeasure (int widthMeasureSpec, int heightMeasureSpec) {
    DisplayMetrics metrics = ctx.getResources().getDisplayMetrics();
    int screenWidth = metrics.widthPixels;
    int screenHeight = metrics.heightPixels;
    int return = Math.min (screenHeight, screenWidth);
    setMeasuredDimension (return, return);
}
```

Custom view 3

- Redefine onDraw():

```
@Override
    protected void onDraw (Canvas canvas) {
        int width = getWidth();
        int step = 10;
        int x = 0;
        int y = 0;
        int size = width;
        boolean switchColor = true;
        while (x <= size) {
            if (switchColor)
                canvas.drawRect(x, y, size, size, mPaintOne);
            else
                canvas.drawRect(x, y, size, size, mPaintTwo);
            x + = step;
            y + = step;
            size - = step;
            switchColor =! switchColor;
        }
    }
```

Custom view 3

- Use a custom view in XML

```
<LinearLayout xmlns: attr = "http://schemas.android.com/apk/res-aut"
        android: layout_width = "match_parent"
        android: layout_height = "match_parent"
        android: orientation = "vertical">
        <com.isi.demoMyView.myView.MyView
            android: layout_width = "match_parent"
            android: layout_height = "match_parent"
            attr: colorTwo = "#5fe177"
            attr: colorOne = "#321b94" />
</ LinearLayout>
```