Graphics

In Android kan je op elke "View" tekenen. Je kan ook een eigen "View" definiëren die afgeleid is van de standaard "View" klasse.

In het voorbeeld werd een "Tekenblad" klasse aangemaakt die afgeleid is van een "View". In de "constructor" functie worden enkele tools ("canvas()", "path()", "paint()") ingesteld om bvb. met jouw vinger op het scherm te kunnen tekenen.

Om dit alles goed te laten werken moet je in de klasse ook nog de functies "onDraw()" en "onTouchEvent()" implementeren. De "drawPath()" functie van het "canvas" object (in de "onDraw()" functie) toont uiteindelijk de uitgevoerde handelingen.


```
package com.android.tekenen;

import android.os.Bundle;
import android.app.Activity;
import android.graphics.Color;
import android.view.Menu;

public class Tekenblok extends Activity {
   Tekenblad papier;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      papier = new Tekenblad(this);
      papier.setBackgroundColor(Color.WHITE);
      setContentView(papier);
      papier.requestFocus();
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.activity_tekenblok, menu);
      return true;
   }
}
```

```
package com.android.tekenen;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.view.MotionEvent;
import android.view.View;

public class Tekenblad extends View {
  private Canvas blad;
  private Paint verf;
  private Path potlood;
  private int kleur;
  private float mx, my, mx1, my1, ra;

  public Tekenblad(Context context) {
    super(context);
    blad = new Canvas();
    potlood = new Path();
    verf = new Paint();
    mx = 30;
    my = 30;
    mx1 = 50;
    my1 =50;
    ra = 30;
  }

  @Override
  public void onDraw(Canvas canvas) {
    kleur = Color.BLACK;
    verf.setColor(kleur);
    verf.setStrokeWidth(5);
    canvas.drawPath(potlood, verf);
  }

  @Override
  public boolean onTouchEvent(MotionEvent event) {
    float x = event.getX();
    float y = event.getY();
    switch (event.getAction()) {
    case MotionEvent.ACTION_DOWN:
       //potlood.reset();
      potlood.moveTo(x, y);
      //potlood.lineTo(x, y);
      mx = x;
      my = y;
      break;
    case MotionEvent.ACTION_MOVE:
      float dx = Math.abs(x - mx);
      float dy = Math.abs(y - my);
      if (dx >= 5 || dy >= 5) {
        //potlood.quadTo(mx, my, (x + mx) / 2, (y + my) / 2);
        potlood.addCircle(mx,my,ra,Path.Direction.CW);
        mx = x;
        my = y;
      }
      break;
    case MotionEvent.ACTION_UP:
      potlood.lineTo(mx, my);
      blad.drawPath(potlood, verf);
       //potlood.reset();
```

```
        break;
      }
      invalidate();
      return true;
    }
  }
```

Er zijn nog verschillende grafische functies waarmee je kan experimenteren: "path.lineTo()", "path.quadTo()", "path.addCircle()", enz…

Je kan de volledige lijst raadplegen op:
http://developer.android.com/reference/android/graphics/Path.html

Met de functie "invalidate()" forceer je de "View" om te tekenen. De handelingen die eraan voorafgingen worden getoond (door de "canvas.drawPath()" functie).

Je kan nog meer informatie vinden op:
http://developer.android.com/training/custom-views/custom-drawing.html

Naast eenvoudige tekeningen kan je ook spelletjes programmeren. Dit valt echter buiten de scope van deze cursus.

Hieronder vind je het "FingerPaint" voorbeeld zoals je dat op het internet kunt vinden. Er zijn enkele extra's voorzien via de menu-ingangen.

TIP: in een AVD device kan je met de F2-toets de menu ophalen.

```
package android.fingerpaint;

import android.app.Activity;
import android.content.Context;
import android.graphics.*;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;

/*
 * Copyright (C) 2007 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
public class FingerPaint extends Activity
        implements ColorPickerDialog.OnColorChangedListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new MyView(this));
```

```
        mPaint = new Paint();
        mPaint.setAntiAlias(true);
        mPaint.setDither(true);
        mPaint.setColor(0xFFFF0000);
        mPaint.setStyle(Paint.Style.STROKE);
        mPaint.setStrokeJoin(Paint.Join.ROUND);
        mPaint.setStrokeCap(Paint.Cap.ROUND);
        mPaint.setStrokeWidth(12);
        mEmboss = new EmbossMaskFilter(new float[] { 1, 1, 1 },
              0.4f, 6, 3.5f);
        mBlur = new BlurMaskFilter(8, BlurMaskFilter.Blur.NORMAL);
    }
    private Paint       mPaint;
    private MaskFilter  mEmboss;
    private MaskFilter  mBlur;
    public void colorChanged(int color) {
        mPaint.setColor(color);
    }
    public class MyView extends View {
        private static final float MINP = 0.25f;
        private static final float MAXP = 0.75f;
        private Bitmap  mBitmap;
        private Canvas  mCanvas;
        private Path    mPath;
        private Paint   mBitmapPaint;
        public MyView(Context c) {
            super(c);
            mPath = new Path();
            mBitmapPaint = new Paint(Paint.DITHER_FLAG);
        }
        @Override
        protected void onSizeChanged(int w, int h, int oldw, int oldh) {
            super.onSizeChanged(w, h, oldw, oldh);
            mBitmap = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);
            mCanvas = new Canvas(mBitmap);
        }
        @Override
        protected void onDraw(Canvas canvas) {
            canvas.drawColor(0xFFAAAAAA);
            canvas.drawBitmap(mBitmap, 0, 0, mBitmapPaint);
            canvas.drawPath(mPath, mPaint);
        }
        private float mX, mY;
        private static final float TOUCH_TOLERANCE = 4;
        private void touch_start(float x, float y) {
            mPath.reset();
            mPath.moveTo(x, y);
            mX = x;
            mY = y;
        }
        private void touch_move(float x, float y) {
            float dx = Math.abs(x - mX);
            float dy = Math.abs(y - mY);
            if (dx >= TOUCH_TOLERANCE || dy >= TOUCH_TOLERANCE) {
                mPath.quadTo(mX, mY, (x + mX)/2, (y + mY)/2);
                mX = x;
                mY = y;
            }
        }
        private void touch_up() {
            mPath.lineTo(mX, mY);
            // commit the path to our offscreen
            mCanvas.drawPath(mPath, mPaint);
```

```
        // kill this so we don't double draw
        mPath.reset();
    }
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        float x = event.getX();
        float y = event.getY();
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                touch_start(x, y);
                invalidate();
                break;
            case MotionEvent.ACTION_MOVE:
                touch_move(x, y);
                invalidate();
                break;
            case MotionEvent.ACTION_UP:
                touch_up();
                invalidate();
                break;
        }
        return true;
    }
}
private static final int COLOR_MENU_ID = Menu.FIRST;
private static final int EMBOSS_MENU_ID = Menu.FIRST + 1;
private static final int BLUR_MENU_ID = Menu.FIRST + 2;
private static final int ERASE_MENU_ID = Menu.FIRST + 3;
private static final int SRCATOP_MENU_ID = Menu.FIRST + 4;
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    menu.add(0, COLOR_MENU_ID, 0, "Color").setShortcut('3', 'c');
    menu.add(0, EMBOSS_MENU_ID, 0, "Emboss").setShortcut('4', 's');
    menu.add(0, BLUR_MENU_ID, 0, "Blur").setShortcut('5', 'z');
    menu.add(0, ERASE_MENU_ID, 0, "Erase").setShortcut('5', 'z');
    menu.add(0, SRCATOP_MENU_ID, 0, "SrcATop").setShortcut('5', 'z');
    return true;
}
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    super.onPrepareOptionsMenu(menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    mPaint.setXfermode(null);
    mPaint.setAlpha(0xFF);
    switch (item.getItemId()) {
        case COLOR_MENU_ID:
            new ColorPickerDialog(this, this, mPaint.getColor()).show();
            return true;
        case EMBOSS_MENU_ID:
            if (mPaint.getMaskFilter() != mEmboss) {
                mPaint.setMaskFilter(mEmboss);
            } else {
                mPaint.setMaskFilter(null);
            }
            return true;
        case BLUR_MENU_ID:
            if (mPaint.getMaskFilter() != mBlur) {
                mPaint.setMaskFilter(mBlur);
            } else {
```

```
                mPaint.setMaskFilter(null);
            }
            return true;
        case ERASE_MENU_ID:
            mPaint.setXfermode(new PorterDuffXfermode(
                    PorterDuff.Mode.CLEAR));
            return true;
        case SRCATOP_MENU_ID:
            mPaint.setXfermode(new PorterDuffXfermode(
                    PorterDuff.Mode.SRC_ATOP));
            mPaint.setAlpha(0x80);
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Met het "ColorPickerDialog" venster kan je een andere kleur kiezen.

```
package android.fingerpaint;

import android.os.Bundle;
import android.app.Dialog;
import android.content.Context;
import android.graphics.*;
import android.view.MotionEvent;
import android.view.View;

/*
 * Copyright (C) 2007 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

public class ColorPickerDialog extends Dialog {

    public interface OnColorChangedListener {
        void colorChanged(int color);
    }

    private OnColorChangedListener mListener;
    private int mInitialColor;

    private static class ColorPickerView extends View {
        private Paint mPaint;
        private Paint mCenterPaint;
        private final int[] mColors;
        private OnColorChangedListener mListener;

        ColorPickerView(Context c, OnColorChangedListener l, int color) {
            super(c);
            mListener = l;
            mColors = new int[] {
                    0xFFFF0000, 0xFFFF00FF, 0xFF0000FF, 0xFF00FFFF, 0xFF00FF00,
```

```
            0xFFFFFF00, 0xFFFF0000
        };
        Shader s = new SweepGradient(0, 0, mColors, null);

        mPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        mPaint.setShader(s);
        mPaint.setStyle(Paint.Style.STROKE);
        mPaint.setStrokeWidth(32);

        mCenterPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        mCenterPaint.setColor(color);
        mCenterPaint.setStrokeWidth(5);
    }

    private boolean mTrackingCenter;
    private boolean mHighlightCenter;

    @Override
    protected void onDraw(Canvas canvas) {
        float r = CENTER_X - mPaint.getStrokeWidth()*0.5f;

        canvas.translate(CENTER_X, CENTER_X);

        canvas.drawOval(new RectF(-r, -r, r, r), mPaint);
        canvas.drawCircle(0, 0, CENTER_RADIUS, mCenterPaint);

        if (mTrackingCenter) {
            int c = mCenterPaint.getColor();
            mCenterPaint.setStyle(Paint.Style.STROKE);

            if (mHighlightCenter) {
                mCenterPaint.setAlpha(0xFF);
            } else {
                mCenterPaint.setAlpha(0x80);
            }
            canvas.drawCircle(0, 0,
                    CENTER_RADIUS + mCenterPaint.getStrokeWidth(),
                    mCenterPaint);

            mCenterPaint.setStyle(Paint.Style.FILL);
            mCenterPaint.setColor(c);
        }
    }

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        setMeasuredDimension(CENTER_X*2, CENTER_Y*2);
    }

    private static final int CENTER_X = 100;
    private static final int CENTER_Y = 100;
    private static final int CENTER_RADIUS = 32;

    private int floatToByte(float x) {
        int n = java.lang.Math.round(x);
        return n;
    }
    private int pinToByte(int n) {
        if (n < 0) {
            n = 0;
        } else if (n > 255) {
            n = 255;
        }
```

```
        return n;
    }

    private int ave(int s, int d, float p) {
        return s + java.lang.Math.round(p * (d - s));
    }

    private int interpColor(int colors[], float unit) {
        if (unit <= 0) {
            return colors[0];
        }
        if (unit >= 1) {
            return colors[colors.length - 1];
        }

        float p = unit * (colors.length - 1);
        int i = (int)p;
        p -= i;

        // now p is just the fractional part [0...1) and i is the index
        int c0 = colors[i];
        int c1 = colors[i+1];
        int a = ave(Color.alpha(c0), Color.alpha(c1), p);
        int r = ave(Color.red(c0), Color.red(c1), p);
        int g = ave(Color.green(c0), Color.green(c1), p);
        int b = ave(Color.blue(c0), Color.blue(c1), p);

        return Color.argb(a, r, g, b);
    }

    private int rotateColor(int color, float rad) {
        float deg = rad * 180 / 3.1415927f;
        int r = Color.red(color);
        int g = Color.green(color);
        int b = Color.blue(color);

        ColorMatrix cm = new ColorMatrix();
        ColorMatrix tmp = new ColorMatrix();

        cm.setRGB2YUV();
        tmp.setRotate(0, deg);
        cm.postConcat(tmp);
        tmp.setYUV2RGB();
        cm.postConcat(tmp);

        final float[] a = cm.getArray();

        int ir = floatToByte(a[0] * r +  a[1] * g +  a[2] * b);
        int ig = floatToByte(a[5] * r +  a[6] * g +  a[7] * b);
        int ib = floatToByte(a[10] * r + a[11] * g + a[12] * b);

        return Color.argb(Color.alpha(color), pinToByte(ir),
                pinToByte(ig), pinToByte(ib));
    }

    private static final float PI = 3.1415926f;

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        float x = event.getX() - CENTER_X;
        float y = event.getY() - CENTER_Y;
        boolean inCenter = java.lang.Math.sqrt(x*x + y*y) <= CENTER_RADIUS;
```

```
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                mTrackingCenter = inCenter;
                if (inCenter) {
                    mHighlightCenter = true;
                    invalidate();
                    break;
                }
            case MotionEvent.ACTION_MOVE:
                if (mTrackingCenter) {
                    if (mHighlightCenter != inCenter) {
                        mHighlightCenter = inCenter;
                        invalidate();
                    }
                } else {
                    float angle = (float)java.lang.Math.atan2(y, x);
                    // need to turn angle [-PI ... PI] into unit [0....1]
                    float unit = angle/(2*PI);
                    if (unit < 0) {
                        unit += 1;
                    }
                    mCenterPaint.setColor(interpColor(mColors, unit));
                    invalidate();
                }
                break;
            case MotionEvent.ACTION_UP:
                if (mTrackingCenter) {
                    if (inCenter) {
                        mListener.colorChanged(mCenterPaint.getColor());
                    }
                    mTrackingCenter = false;    // so we draw w/o halo
                    invalidate();
                }
                break;
        }
        return true;
    }
}

public ColorPickerDialog(Context context,
                         OnColorChangedListener listener,
                         int initialColor) {
    super(context);

    mListener = listener;
    mInitialColor = initialColor;
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    OnColorChangedListener l = new OnColorChangedListener() {
        public void colorChanged(int color) {
            mListener.colorChanged(color);
            dismiss();
        }
    };

    setContentView(new ColorPickerView(getContext(), l, mInitialColor));
    setTitle("Pick a Color");
}
}
```