Trường Đại học Bách Khoa Hà Nội
Hanoi University of Science and Technology

Chapter 12. Graphics 2D

Hanoi University of Science and Technology

Nguyễn Hồng Quang
19/8/2012

Nội dung

- Color, Paint, Canvas, Path
- Ví dụ : Sudoku

Android Graphics library

- Android provides a complete native two-dimensional graphics library in its android.graphics package.
- Color, Paint, Canvas, Path

Color

- Android colors are represented with four numbers, one each for alpha, red, green, and blue (ARGB).
- Each component can have 256 possible values, or 8 bits, so a color is typically packed into a 32-bit integer.

Color
Alpha

- *Alpha is a measure of transparency. The lowest value, 0, indicates the* color is completely transparent.
- It doesn't really matter what the values for RGB are, if A is 0.
- The highest value, 255, indicates the color is completely opaque.
- Values in the middle are used for translucent, or semitransparent, colors. They let you see some of what is underneath the object being drawn in the foreground.

Create a color

- Direct:

```
int color = Color.BLUE; // solid blue
// Translucent purple
color = Color.argb(127, 255, 0, 255);
```

- Defining colors in an XML resource file

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<color name="mycolor">#7fff00ff</color>
</resources>
color = getResources().getColor(R.color.mycolor);
```

Paint

- It holds the style, color, and other information needed to draw any graphics including bitmaps, text, and geometric shapes.
- Examples:

```
Paint cPaint = new Paint();
cPaint.setColor(Color.LTGRAY);
```

Canvas

- The Canvas class represents a surface on which you draw.
- Methods on the Canvas class let you draw lines, rectangles, circles, or other arbitrary graphics on the surface.
- In Android, the display screen is taken up by an Activity, which hosts a View, which in turn hosts a Canvas. You get an opportunity to draw on that canvas by overriding the View.onDraw( ) method. The only parameter to onDraw( ) is a canvas on which you can draw.

Example activity

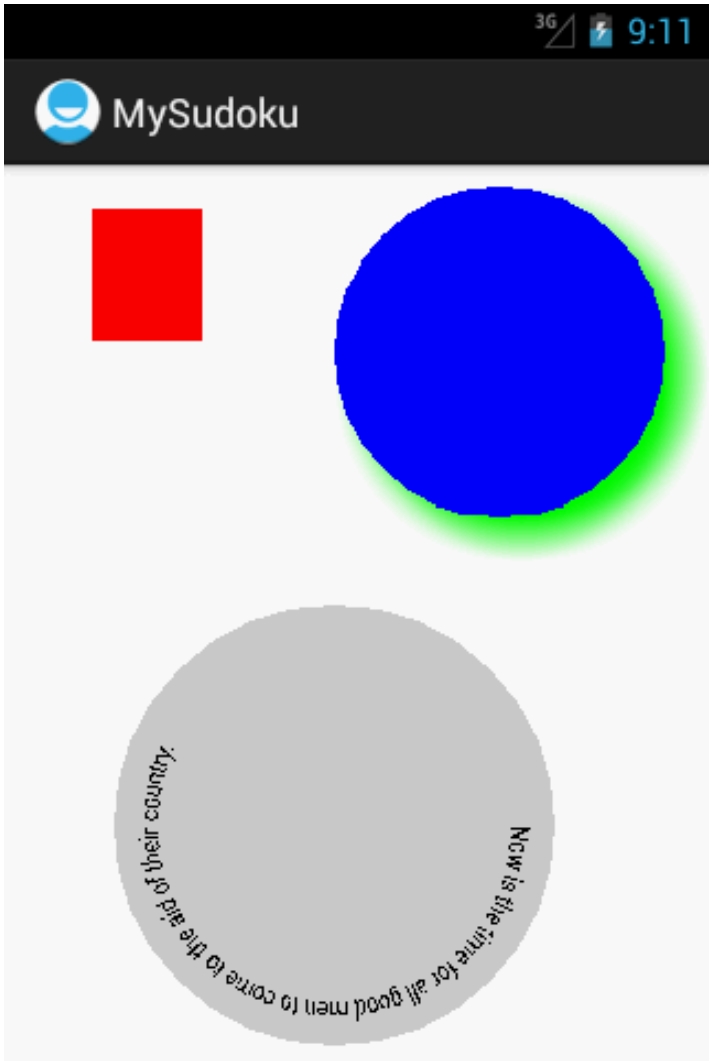
```
public class Graphics extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new GraphicsView(this));
    }
    static public class GraphicsView extends View {
      …
    }
}
```

Example activity

```
static public class GraphicsView extends View {
    public GraphicsView(Context context) {
        super(context);
    }
    @Override
    protected void onDraw(Canvas canvas) {
        // Drawing commands go here
    }
}
```

Các hàm vẽ

```
// this is the "paintbrush"
Paint paint = new Paint();
// set the color
paint.setColor(Color.RED);

// draw Rectangle with corners at (40, 20) and (90, 80)
canvas.drawRect(40, 20, 90, 80, paint);

// change the color
paint.setColor(Color.BLUE);
// set a shadow
paint.setShadowLayer(10, 10, 10, Color.GREEN);

// create a "bounding rectangle"
RectF rect = new RectF(150, 10, 300, 160);

// draw an oval in the bounding rectangle
canvas.drawOval(rect, paint);
```

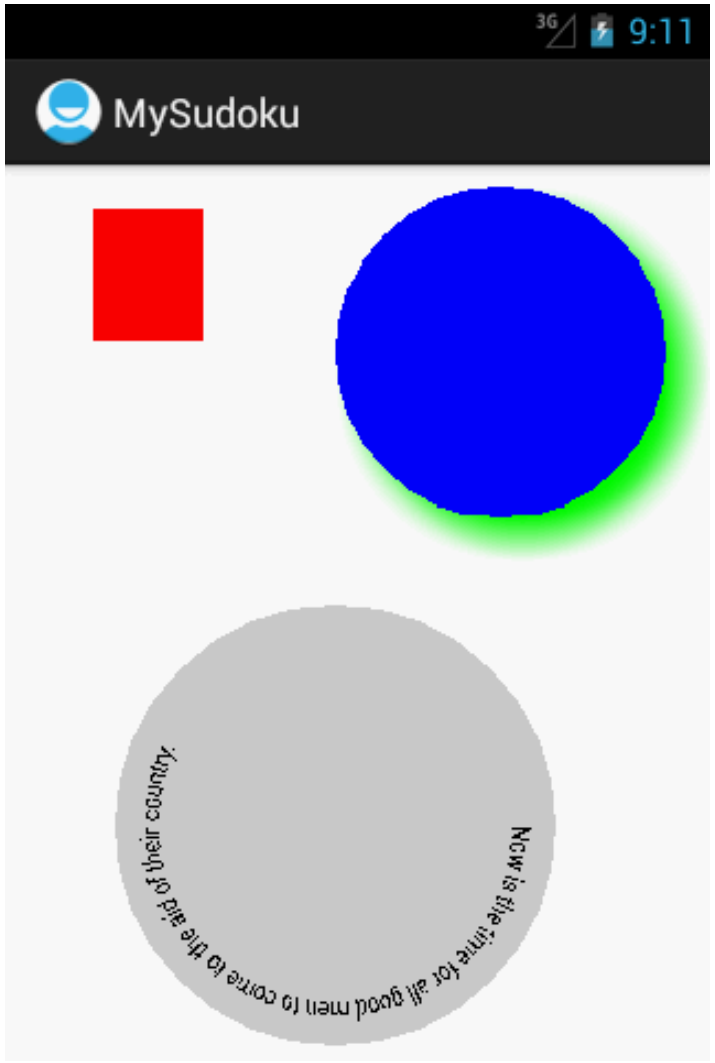


Path

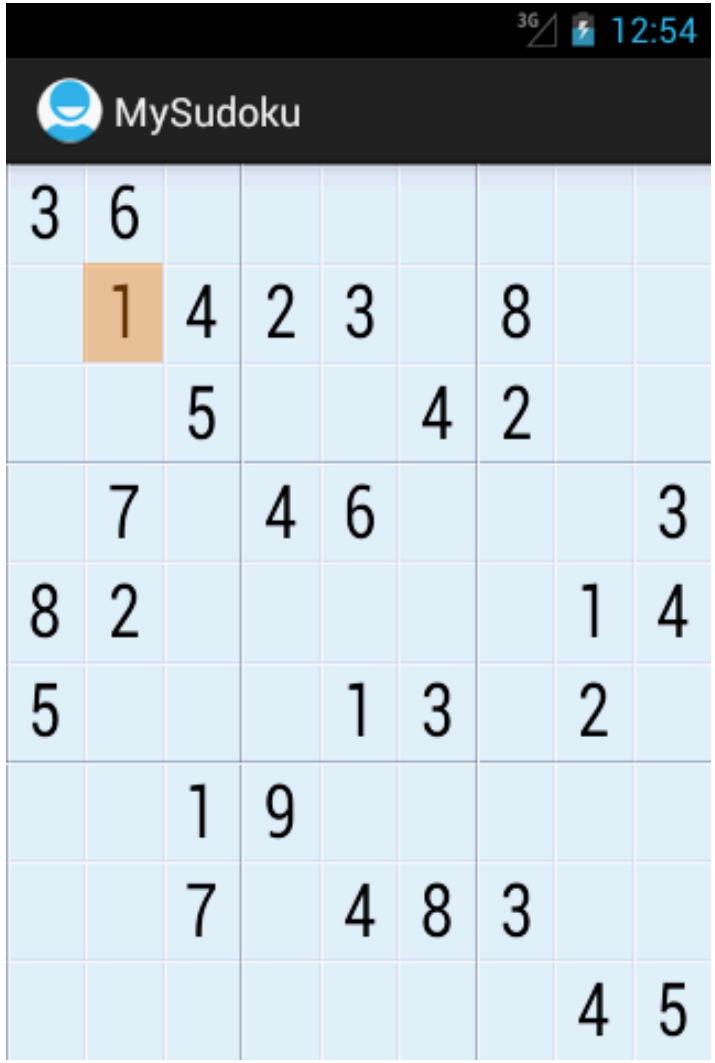
- The Path class holds a set of vector-drawing commands such as lines, rectangles, and curves.

Các hàm vẽ

```
Path circle = new Path();
circle.addCircle(150, 300, 100, Direction.CW);

String QUOTE = "Now is the time for all " +
"good men to come to the aid of their country." ;

Paint cPaint = new Paint();
cPaint.setColor(Color.LTGRAY);
Paint tPaint = new Paint();
tPaint.setColor(Color.BLACK);

canvas.drawPath(circle, cPaint);
canvas.drawTextOnPath(QUOTE, circle, 0, 20,
    tPaint);
```



Game Sudoku



Step 1. Main Activity

```
public class MySudoku extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.main);
        setContentView(new DrawableView(this));
    }
}
```

Step 2. colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="puzzle_background">#ffe6f0ff</color>
    <color name="puzzle_hilite">#ffffffff</color>
    <color name="puzzle_light">#64c6d4ef</color>
    <color name="puzzle_dark">#6456648f</color>
    <color name="puzzle_foreground">#ff000000</color>
    <color name="puzzle_hint_0">#64ff0000</color>
    <color name="puzzle_hint_1">#6400ff80</color>
    <color name="puzzle_hint_2">#2000ff80</color>
    <color name="puzzle_selected">#64ff8000</color>
</resources>
```

Step 3. DrawableView Activity Class variables

```
private final String easyPuzzle =
    "360000000004230800000004200" +
    "070460003820000014500013020" +
    "001900000007048300000000045" ;
private int puzzle[];

private float width; // width of one tile
private float height; // height of one tile
private int selX; // X index of selection
private int selY; // Y index of selection
private final Rect selRect = new Rect();

Context context;
```

Step 3. DrawableView Activity Data model

```
static protected int[] fromPuzzleString(String string) {
    int[] puz = new int[string.length()];
    for (int i = 0; i < puz.length; i++) {
        puz[i] = string.charAt(i) - '0' ;
    }
    return puz;
}
```

Step 3. DrawableView Activity Data model

```
/** Return the tile at the given coordinates */
private int getTile(int x, int y) {
    return puzzle[y * 9 + x];
}

/** Change the tile at the given coordinates */
private void setTile(int x, int y, int value) {
    puzzle[y * 9 + x] = value;
}
```

Step 3. DrawableView Activity Data model

```
/** Return a string for the tile at the given coordinates */
protected String getTileString(int x, int y) {
    int v = getTile(x, y);
    if (v == 0)
        return "" ;
    else
        return String.valueOf(v);
}
```

Step 4. DrawableView Activity Constructor

```
public DrawableView(Context context) {
    super(context);
    // TODO Auto-generated constructor stub
    puzzle = fromPuzzleString(easyPuzzle);
    setFocusable(true);
    setFocusableInTouchMode(true);
    this.context = context;
}
```

Xác định kích thước màn hình

```
@Override
protected void onSizeChanged(int w, int h, int oldw,
    int oldh) {
    super.onSizeChanged(w, h, oldw, oldh);
    width = w / 9f;
    height = h / 9f;
    getRect(selX, selY, selRect);
    Log.d("MyLog", "onSizeChanged: width " + width + ",
        height " + height);
}
```

Step 5. DrawableView Activity onDraw

```
protected void onDraw(Canvas canvas) {
    // TODO Auto-generated method stub
    super.onDraw(canvas);
    // Draw the background...
    Paint background = new Paint();
    background.setColor(
        getResources().getColor(R.color.puzzle_background));
    canvas.drawRect(0, 0, getWidth(), getHeight(), background);

    // Draw the board...
    // Draw the numbers...
    // Draw the selection...
}
```

Step 5a. DrawableView Activity Draw the board...

```
// Define colors for the grid lines
Paint dark = new Paint();
dark.setColor(getResources().getColor(R.color.puzzle_dark));
Paint hilite = new Paint();
hilite.setColor(getResources().getColor(R.color.puzzle_hilite));
Paint light = new Paint();
light.setColor(getResources().getColor(R.color.puzzle_light));
```

Step 5a. DrawableView Activity Draw the board...

```
// Draw the minor grid lines
for (int i = 0; i < 9; i++) {
    canvas.drawLine(0, i * height, getWidth(), i * height,
        light);
    canvas.drawLine(0, i * height + 1, getWidth(), i * height
        + 1, hilite);
    canvas.drawLine(i * width, 0, i * width, getHeight(),
        light);
    canvas.drawLine(i * width + 1, 0, i * width + 1,
        getHeight(), hilite);
}
```

Step 5a. DrawableView Activity Draw the board...

```
// Draw the major grid lines
for (int i = 0; i < 9; i++) {
    if (i % 3 != 0)
        continue;
    canvas.drawLine(0, i * height, getWidth(), i * height, dark);
    canvas.drawLine(0, i * height + 1, getWidth(), i * height + 1, hilite);
    canvas.drawLine(i * width, 0, i * width, getHeight(), dark);
    canvas.drawLine(i * width + 1, 0, i * width + 1, getHeight(), hilite);
}
```

Step 5b. DrawableView Draw the numbers...

```
// Define color and style for numbers
Paint foreground = new Paint(Paint.ANTI_ALIAS_FLAG);
foreground.setColor(getResources().getColor(R.color.puzzle_foreground));
//foreground.setStyle(Style.FILL);
foreground.setTextSize(height * 0.75f);
foreground.setTextScaleX(width / height);
foreground.setTextAlign(Paint.Align.CENTER);
```

Step 5b. DrawableView Draw the numbers...

```
// Draw the number in the center of the tile
FontMetrics fm = foreground.getFontMetrics();
// Centering in X: use alignment (and X at midpoint)
float x = width / 2;
// Centering in Y: measure ascent/descent first
float y = height / 2 - (fm.ascent + fm.descent) / 2;
for (int i = 0; i < 9; i++) {
    for (int j = 0; j < 9; j++) {
        canvas.drawText(getTileString(i, j),  i * width + x,
          j * height + y, foreground);
    }
}
```

Step 5c. DrawableView Activity Draw the selection...

```
Log.d("MyLog", "selRect=" + selRect);
Paint selected = new Paint();
selected.setColor(getResources().getColor(R.color.puzzle_selected));
canvas.drawRect(selRect, selected);
```

Step 6a. Select

```
private void select(int x, int y) {
    invalidate(selRect);
    selX = Math.min(Math.max(x, 0), 8);
    selY = Math.min(Math.max(y, 0), 8);
    getRect(selX, selY, selRect);
    invalidate(selRect);
}
```

Step 6b. TouchEvent

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() !=
       MotionEvent.ACTION_DOWN)
    return super.onTouchEvent(event);
    select((int) (event.getX() / width), (int)
       (event.getY() / height));
    Log.d("MyLog", "onTouchEvent: x " + selX + ", y " +
       selY);
    return true;
}
```

Step 6c. KeyDown event

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent
    event) {
    //Log.d("MyLog", "onKeyDown: keycode=" + keyCode + ",
    event=" + event);
     switch (keyCode) {
     case KeyEvent.KEYCODE_1:
          setTile(selX, selY, 1);
          invalidate();
          break;
```

Step 6c. KeyDown event

```
case KeyEvent.KEYCODE_DPAD_UP:
    select(selX, selY - 1);
    break;
case KeyEvent.KEYCODE_DPAD_DOWN:
    select(selX, selY + 1);
    break;
case KeyEvent.KEYCODE_DPAD_LEFT:
    select(selX - 1, selY);
    break;
case KeyEvent.KEYCODE_DPAD_RIGHT:
    select(selX + 1, selY);
    break;
```

Step 7. Animation
Continue in onKeyDown method

```
default:
    //return super.onKeyDown(keyCode, event);
    Log.d("MyLog", "setSelectedTile: invalid: " + keyCode);
    startAnimation(AnimationUtils.loadAnimation(context,
    R.anim.shake));
```

Animation res/anim/cycle_7.xml

```
<?xml version="1.0" encoding="utf-8"?>
<cycleInterpolator
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:cycles="7" />
```

Animation res/anim/shake.xml

```
<?xml version="1.0" encoding="utf-8"?>
<translate
    xmlns:android="http://schemas.android.com/apk/res/andr
    oid"
    android:fromXDelta="0"
    android:toXDelta="10"
    android:duration="1000"
    android:interpolator="@anim/cycle_7" />
```

End of Lecture

Q&A