Handheld Application Development

Lec 10-11: Multimedia I & II

Ekarat Rattagan, Ph.D.

Outline

- 2D graphics
- Touch events
- Animation

Android Graphics Programming

- There are many ways to do graphics programming in Android
    - 2D VS 3D
    - Static VS Dynamic
- Many of them require a lot of knowledge of the underlying graphics libraries
- We will look at the very simplest from of 2D graphics

Drawing on a Canvas

- Each **View** has an associated **Canvas**

- When the **View** is shown, its **onDraw** method is automatically called by Android system

- It uses the **Canvas** to render the different things it wants to display

- We can create our **own View** with our **own onDraw** method to display basic objects using the **Canvas**

Canvas and Paint

- Canvas has methods for drawing **Arcs, Bitmaps, Circles, Lines, Ovals, Paths, Rectangles**, etc.

  - Also methods to **rotate, scale, skew, translate**

- **Paint** has methods for setting the alpha, color, shade, stroke, etc.

Canvas and Paint

- Screen x,y coordinate

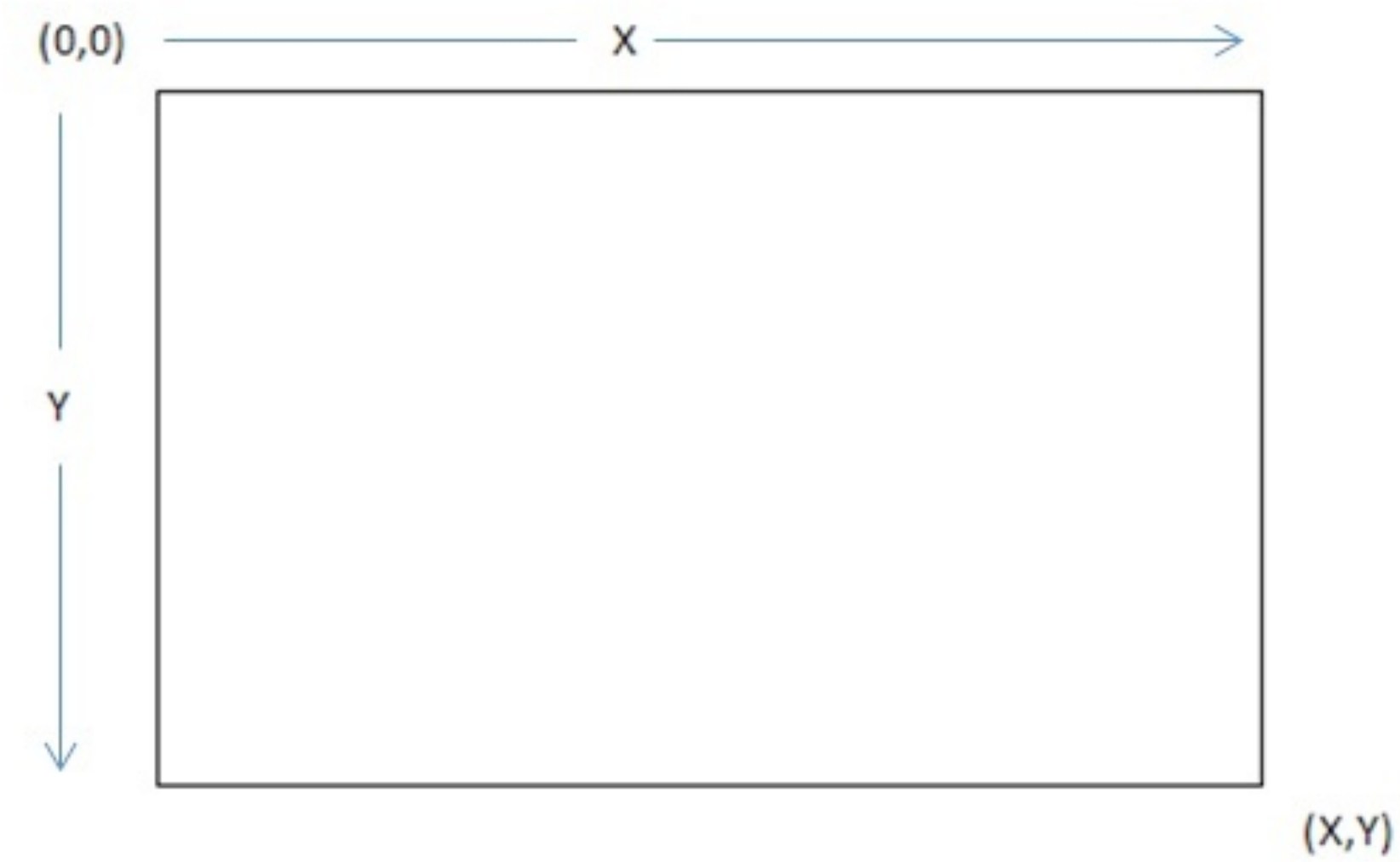

Figure 1 - Canvas coordinates

Create your own View

```
1 package [your package];
2
3 public class DrawableView extends View {
4
5    // Second, you must implement these constructors!!
6    public DrawableView(Context c) {
7        super(c);
8    }
9    public DrawableView(Context c, AttributeSet a) {
10       super(c, a);
11   }
12
```

... continued on next slide ...

Create your own View

Still in the DrawableView class...

```
13    // Third, implement the onDraw method.
14    // This method is called when the View is displayed
15    protected void onDraw(Canvas canvas) {
16
17        // this is the "paintbrush"
18        Paint paint = new Paint();
19        // set the color
20        paint.setColor(Color.RED);
21
22        // draw Rectangle with corners at (40, 20) and (90, 80)
23        canvas.drawRect(40, 20, 90, 80, paint);
24
25        // change the color
26        paint.setColor(Color.BLUE);
27        // set a shadow
28        paint.setShadowLayer(10, 10, 10, Color.GREEN);
29
30        // create a "bounding rectangle"
31        RectF rect = new RectF(150, 150, 280, 280);
32        // draw an oval in the bounding rectangle
33        canvas.drawOval(rect, paint);
34    }
35
36 } // end of DrawableView class
```

Canvas methods

```
c.drawARGB(alpha,r,g,b);                    //fill window with color (rgb=0-255)
c.drawArc(…);                                    //draw a partial ellipse
c.drawBitmap(bmp, x, y, null);                   //draw an image
c.drawCircle(centerX,centerY,r,paint);           //draw a circle
c.drawLine(x1,y1,x2,y2,paint);                   //draw a line segment
c.drawOval(x1, y1, x2, y2, paint);               * (requires Android 5.0)
c.drawOval(newRectF(x1,y1,x2,y2),paint);         //draw oval / circle
c.drawPoint(x,y,paint);                          //color a single pixel
c.drawRect(x1,y1,x2,y2,paint);                   *(requiresAndroid5.0)
c.drawRect(newRectF(x1,y1,x2,y2),paint);         //draw rectangle
c.drawRoundRect(x1,y1,x2,y2,rx,ry,paint);        *(requiresAndroid5.0)
c.drawRoundRect(newRectF(x1,y1,x2,y2),rx,ry,paint);
c.drawText(“str",x,y,paint);                     //draw a text string
c.getWidth(),c.getHeight();                      //get dimensions of drawing
```

Typeface

In Android, **a font is called a Typeface.** Set a font inside a Paint. You can create a Typeface based on a specific font name:

Typeface.create("font name", Typeface.STYLE)

styles: NORMAL, BOLD, ITALIC, BOLD_ITALIC

- Or based on a general "font family”:

  Typeface.create(Typeface.FAMILY_NAME, Typeface.STYLE)

  family names: DEFAULT, MONOSPACE, SERIF, SANS_SERIF

- Or from a file in your src/main/assets/ directory:

  Typeface.createFromAsset(getAssets( ), "filename")

```
// example: use a 40-point monospaced blue font
Paint p = new Paint();
p.setTypeface(Typeface.create(Typeface.MONOSPACE, Typeface.BOLD));
p.setTextSize(40);
p.setARGB(255, 0, 0, 255);
```

Touch events

Touch Event

- When the user touches/clicks on the View, Android invokes the View's **onTouchEvent** method
- A **MotionEvent** object is automatically generated and is passed to the method
- From the MotionEvent, you can determine:
  - type of Action (down, up/release, move)
  - (x,y) coordinate
  - Time when event occurred

Touch event example

1. Modify onDraw so that the color of the rectangle is randomized
2. Adding an onTouchEvent method that looks for an “up” action and calls **invalidate** if the touch is within the bounds of the rectanble.

Modify onDraw as follows

```
// This version of onDraw randomly chooses a color
// to use when drawing the rectangle
protected void onDraw(Canvas canvas) {

    // this is the "paintbrush"
    Paint paint = new Paint();

    // set the color randomly
    int whichColor = (int)(Math.random() * 4);
    if (whichColor == 0) paint.setColor(Color.RED);
    else if (whichColor == 1) paint.setColor(Color.GREEN);
    else if (whichColor == 2) paint.setColor(Color.BLUE);
    else paint.setColor(Color.YELLOW);

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


}
```

Add an onTouchEvent method

```
// this method is called when the user touches the View
public boolean onTouchEvent(MotionEvent event) {

    // if it's an up (“release”) action
    if (event.getAction() == MotionEvent.ACTION_UP) {

        // get the coordinates
        float x = event.getX();
        float y = event.getY();

        // see if they clicked on the box
        if (x >= 40 && x <= 90 && y >= 20 && y <= 80) {

            // redraw the View... this calls onDraw again!
            invalidate();
        }
    }
    // indicates that the event was handled
    return true;
  } // end of onTouchEvent
} // end of DrawableView class
```

Animation

- We use **Handler** to control animation thread
- A Handler allows you to send and process Message and Runnable objects associated with a thread's MessageQueue.
- There are two main uses for a Handler:
    1. to schedule messages and runnables to be executed as some point in the future; and
    2. to enqueue an action to be performed on a different thread than your own.
- Scheduling messages is accomplished with the post(Runnable), postAtTime(Runnable, long),postDelayed(Runnable, Object, long), sendEmptyMessage(int), sendMessage(Message), sendMessageAtTime(Message, long), and sendMessageDelayed(Message, long) methods.

Example of Animation

```
Handler h;

public DrawableView(Context context)  {  super(context);  }
public DrawableView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        h = new Handler();
}
Runnable r = new Runnable() {
        @Override
        public void run() {
                invalidate(); //Redraw the canvas
         }
};

int x = 0;
@Override
protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawRect( 200+x, 200, 400+x, 400, paint);
        x+=10;
        h.postDelayed(r,50); //Update every 50 milliseconds
}
```