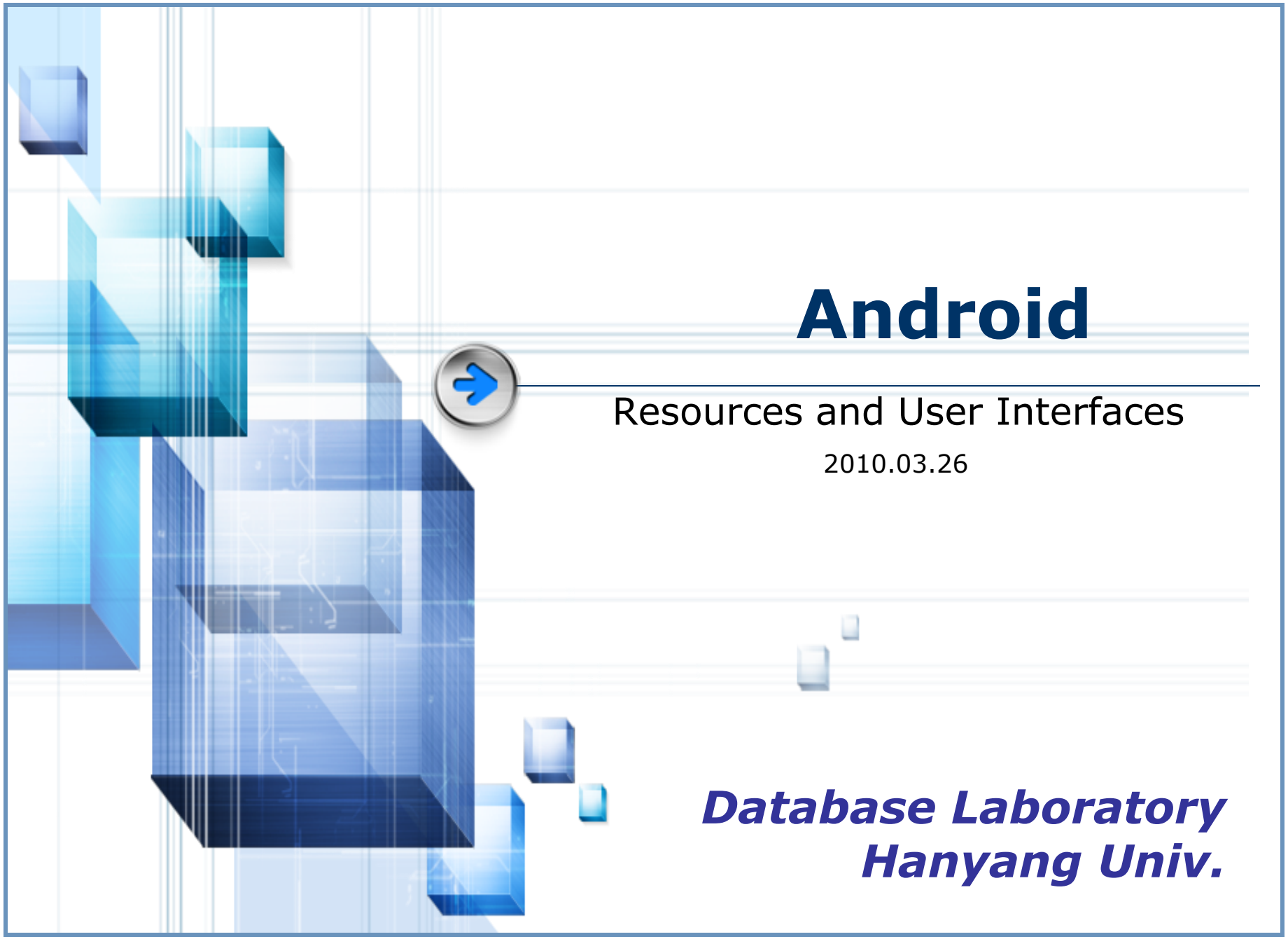

Android

Resources and User Interfaces

2010.03.26

Database Laboratory
Hanyang Univ.

Creating a Compass View(1/11)

- **In the following example, you'll create a new Compass View by extending the View class**

1. **Create a new Compass project that will contain your new Compass View and an Activity to hold it.**
   - Create constructors that will allow the View to be instantiated in code, or through inflation from a resource layout
   - Add a new init *CompassView* method that will be used to initialize the control and call it from each constructor.

Creating a Compass View(2/11)

```
package com.paad.compass;

import android.content.Context;
import android.content.res.Resources;
import android.graphics.*;
import android.util.AttributeSet;
import android.view.*;

public class CompassView extends View{

    public CompassView(Context context){
        super(context);
        initCompassView();
    }

    public CompassView(Context context, AttributeSet attrs) {
        super(context, attrs);
        initCompassView();
    }

    public CompassView(Context context, AttributeSet ats, int defaultStyle){
        super(context, ats, defaultStyle);
        initCompassView();
    }

    protected void initCompassView() {
        setFocusable(true);
    }
}
```

Creating a Compass View(3/11)

2. **The compass control should always be a perfect circle that takes up as much of the canvas as this restriction allows**
   - Override the *onMeasure* method to calculate the length of the shortest side
   - use *setMeasuredDimensio*n to set the height and width using this value

Creating a Compass View(4/11)

```
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    //The compass is a circle that fills as much space as possible.
    //Set the measured dimensions by figuring out the shortest boundary,
    //height or width.
    int measureWidth = measure(widthMeasureSpec);
    int measureHeigth = measure(heightMeasureSpec);

    int d = Math.min(measureWidth, measureHeigth);

    setMeasuredDimension(d, d);
}

private int measure(int measureSpec) {
    int result = 0;

    //Decode the measurement specifications.
    int specMode = MeasureSpec.getMode(measureSpec);
    int specSize = MeasureSpec.getSize(measureSpec);

    if(specMode == MeasureSpec.UNSPECIFIED) {
    //Return a default size of 200 if no bounds are specified.
        result = 200;
    } else {
        //As you want to fill the available space
        //always return the full available bounds.
        result = specSize;
    }
    return result;
}
```

Creating a Compass View(5/11)

3. Create two new resource fi les that store the colors and text strings you'll use to draw the compass

1) Create the text string resource res/values/strings.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">Compass</string>
  <string name="cardinal_north">N</string>
  <string name="cardinal_east">E</string>
  <string name="cardinal_south">S</string>
  <string name="cardinal_west">W</string>
</resources>
```

2) Create the color resource res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="background_color">#F555</color>
  <color name="marker_color">#AFFF</color>
  <color name="text_color">#AFFF</color>
</resources>
```

Creating a Compass View(6/11)

4. **Now return to the *CompassView* class. Add a new property for the bearing to display and create get and set methods for it**

```
private float bearing;

public void setBearing(float _bearing) {
    bearing = _bearing;
}

public float getBearing() {
    return bearing;
}
```

5. **Next, return to the *initCompassView* method, and get references to each resource created in Step 3**
   - Store the String values as instance variables
   - use the color values to create new class-scoped Paint objects
   - You'll use these objects in the next step to draw the compass face

Creating a Compass View(7/11)

```
private Paint markerPaint;
private Paint textPaint;
private Paint circlePaint;
private String northString;
private String southString;
private String eastString;
private String westString;
private int textHeight;

protected void initCompassView() {
    setFocusable(true);

    Resources r = this.getResources();

    circlePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    circlePaint.setColor(r.getColor(R.color.background_color));
    circlePaint.setStrokeWidth(1);
    circlePaint.setStyle(Paint.Style.FILL_AND_STROKE);

    northString = r.getString(R.string.cardinal_north);
    eastString = r.getString(R.string.cardinal_east);
    southString = r.getString(R.string.cardinal_south);
    westString = r.getString(R.string.cardinal_west);

    textPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    textPaint.setColor(r.getColor(R.color.text_color));

    textHeight = (int)textPaint.measureText("yY");

    markerPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    markerPaint.setColor(r.getColor(R.color.marker_color));
}
```

Creating a Compass View(8/11)

6. **The final step is drawing the compass face using the Strings and Paint objects you created in Step 5**
   1) Start by overriding the onDraw method in the *CompassView* class

```
@Override
protected void onDraw(Canvas canvas) {
```

   2) Find the center of the control, and store the length of the smallest side as the compass's radius.

```
int px = getMeasuredWidth() / 2;
int py = getMeasuredHeight() / 2;

int radius = Math.min(px, py);
```

   3) Draw the outer boundary, and color the background of the compass face using the *drawCircle* method. Use the *circlePaint* object you created in Step 5

```
// Draw the background
canvas.drawCircle(px, py, radius, circlePaint);
```

Creating a Compass View(9/11)

4) This compass displays the current heading by rotating the face, so that the current direction is always at the top of the device. To do this, rotate the canvas in the opposite direction to the current heading.

```
// Rotate our perspective so that the 'top' is facing the current bearing.
canvas.save();
canvas.rotate(-bearing, px, py);
```

5) Rotate the canvas through a full rotation, drawing markings every 15 degrees and the abbreviated direction string every 45 degrees

```
int textWidth = (int)textPaint.measureText("W");
int cardinalX = px-textWidth/2;
int cardinalY = py-radius+textHeight;

// Draw the marker every 15 degrees and a text every 45.
for (int i = 0; i < 24; i++) {
  // Draw a marker.
  canvas.drawLine(px, py-radius, px, py-radius+10, markerPaint);

  canvas.save();
  canvas.translate(0, textHeight);

  // Draw the cardinal points
  if (i % 6 == 0) {
    String dirString = "";
    switch (i) {
      case(0)   : {
```

Creating a Compass View(10/11)

```
                        dirString = northString;
                        int arrowY = 2*textHeight;
                        canvas.drawLine(px, arrowY, px-5, 3*textHeight, markerPaint);
                        canvas.drawLine(px, arrowY, px+5, 3*textHeight, markerPaint);
                        break;
                      }
          case(6)  : dirString = eastString; break;
          case(12) : dirString = southString; break;
          case(18) : dirString = westString; break;
        }

        canvas.drawText(dirString, cardinalX, cardinalY, textPaint);
      }

      else if (i % 3 == 0) {
        // Draw the text every alternate 45deg
        String angle = String.valueOf(i*15);
        float angleTextWidth = textPaint.measureText(angle);

        int angleTextX = (int)(px-angleTextWidth/2);
        int angleTextY = py-radius+textHeight;
        canvas.drawText(angle, angleTextX, angleTextY, textPaint);
      }
      canvas.restore();
      canvas.rotate(15, px, py);
    }

    canvas.restore();
}
```

Creating a Compass View(11/11)

6) To view the compass, modify the main.xml layout resource and replace the *TextView* reference with your new *CompassView*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <com.paad.compass.CompassView
    android:id="@+id/compassView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
  />
</LinearLayout>
```

7) Run the Activity, and you should see the *CompassView* displayed