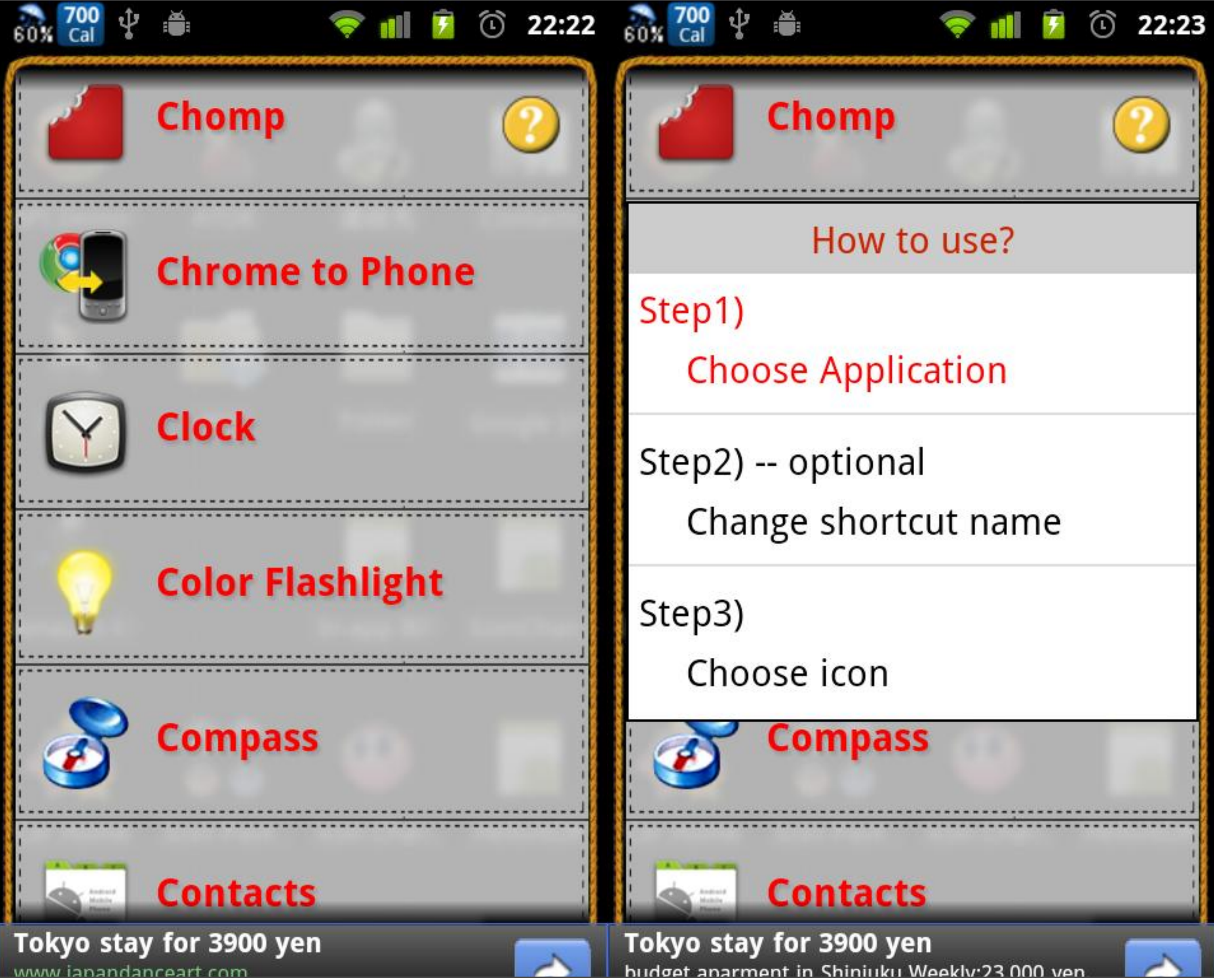
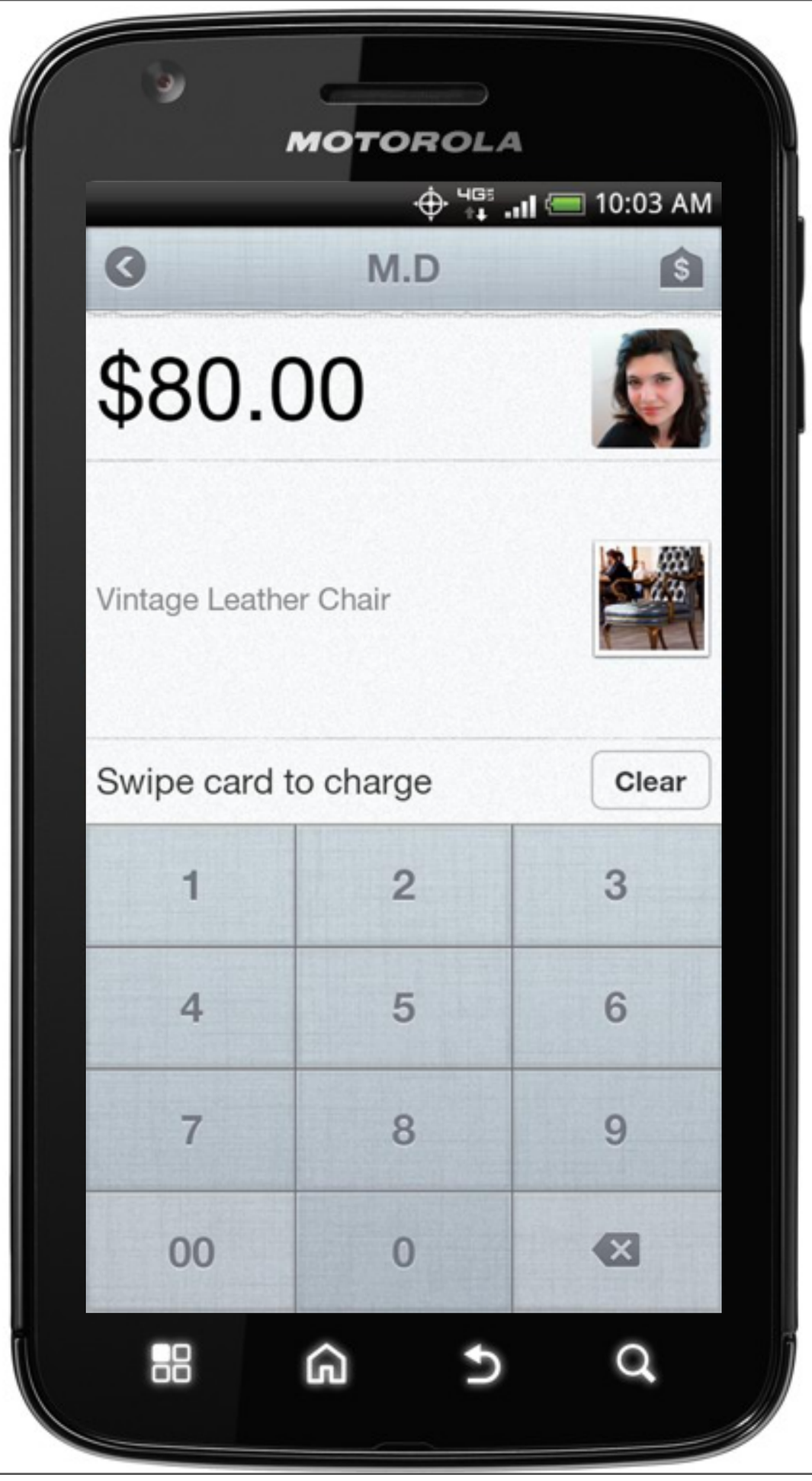
Taming Android

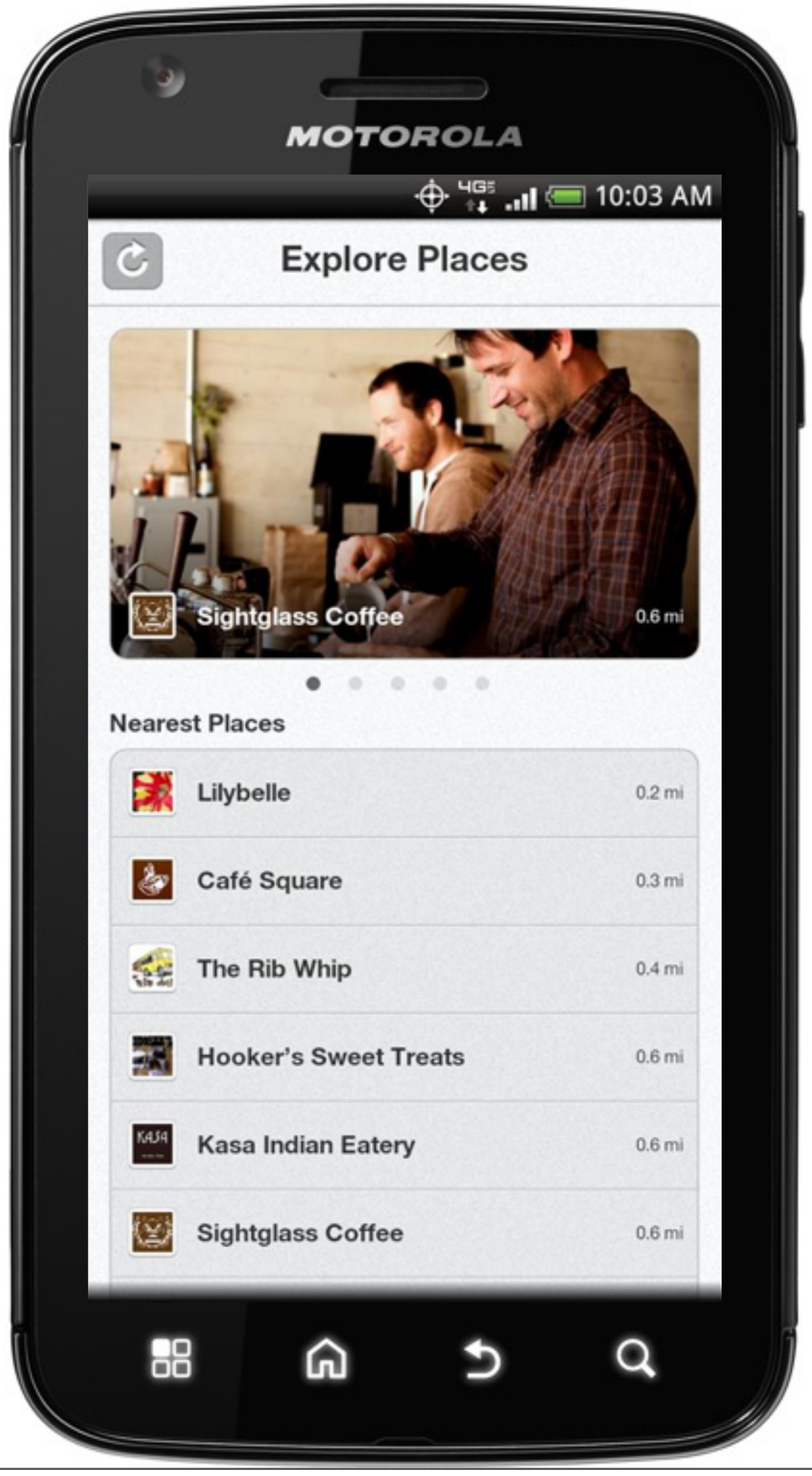
by Eric Burke
Square Inc.

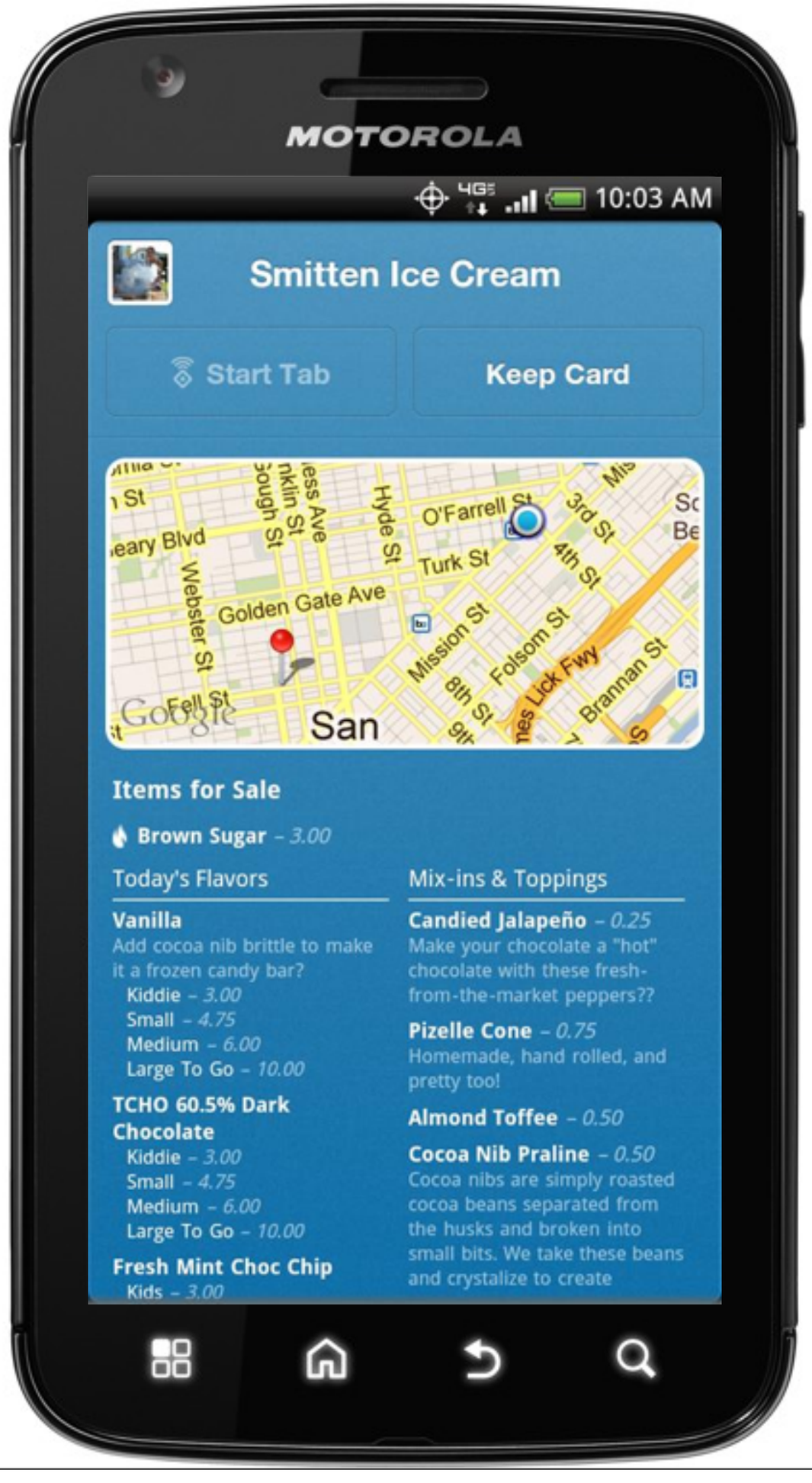
Android Developers?

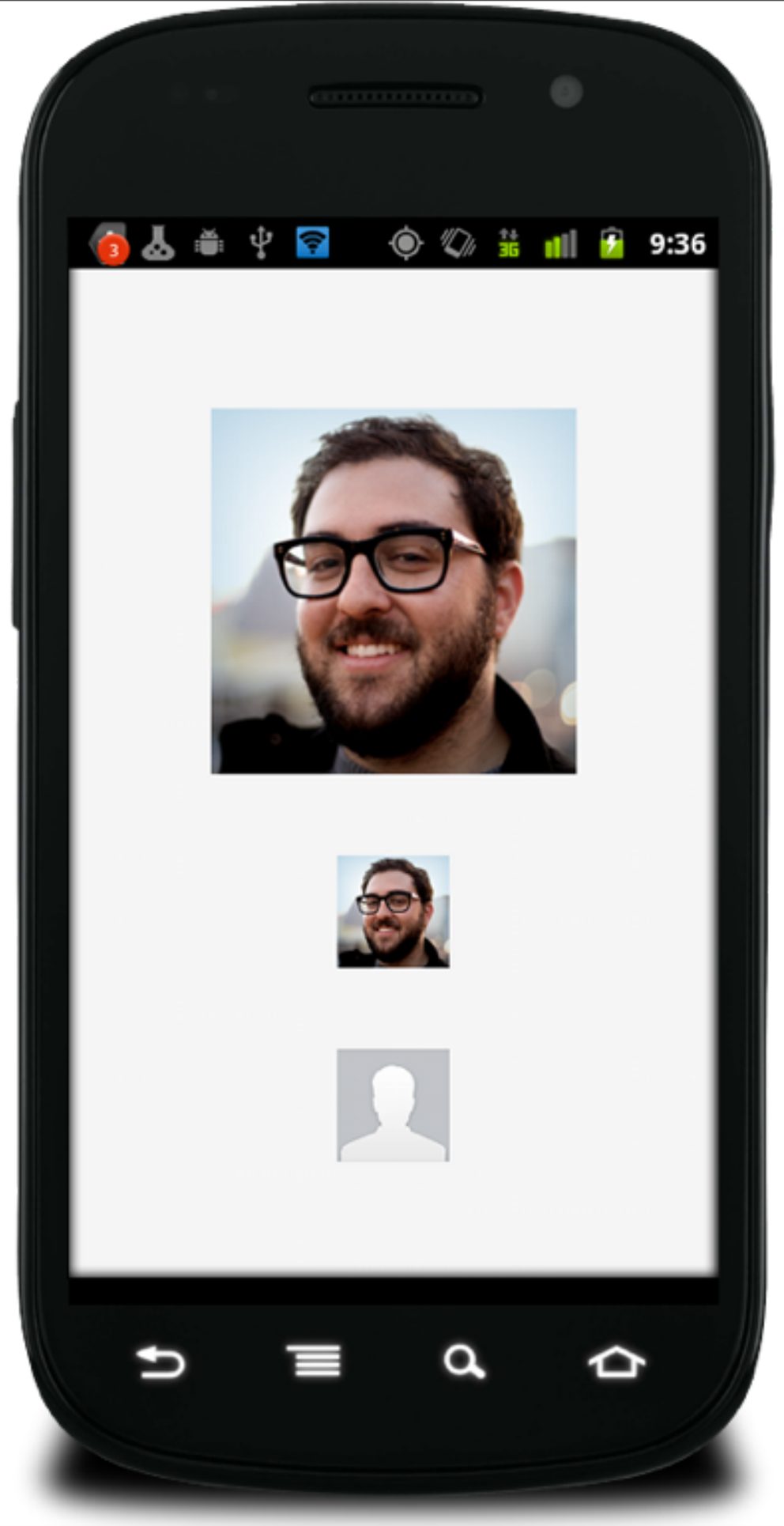

60%
700 Cal
22:22
Chomp
Chrome to Phone
Clock
Color Flashlight
Compass
Contacts
Tokyo stay for 3900 yen
www.japandanceart.com
60%
700 Cal
22:23
Chomp
How to use?
Step1)
Choose Application
Step2) -- optional
Change shortcut name
Step3)
Choose icon
Compass
Contacts
Tokyo stay for 3900 yen
budget aparment in Shinjuku Weekly:23,000 yen


MOTOROLA
10:03 AM
M.D
Maja Daley
Sightglass Coffee
Toasty Melts
Smitten Ice Cream
The Rib Whip
Explore Places


MOTOROLA
10:03 AM
M.D
$80.00
Vintage Leather Chair
Swipe card to charge
Clear
1
2
3
4
5
6
7
8
9
00
0


MOTOROLA
10:03 AM
Explore Places
Sightglass Coffee
0.6 mi
Nearest Places
Lilybelle
0.2 mi
Café Square
0.3 mi
The Rib Whip
0.4 mi
Hooker's Sweet Treats
0.6 mi
Kasa Indian Eatery
0.6 mi
Sightglass Coffee
0.6 mi


MOTOROLA
10:03 AM
Smitten Ice Cream
Start Tab
Keep Card
Golden Gate Ave
Webster St
Hyde St
Turk St
O'Farrell St
Mission St
Folsom St
4th St
3rd St
Brannan St
Fell St
8th St
Google
San
Items for Sale
Brown Sugar – 3.00
Today's Flavors
Vanilla
Add cocoa nib brittle to make it a frozen candy bar?
Kiddie – 3.00
Small – 4.75
Medium – 6.00
Large To Go – 10.00
TCHO 60.5% Dark Chocolate
Kiddie – 3.00
Small – 4.75
Medium – 6.00
Large To Go – 10.00
Fresh Mint Choc Chip
Kids – 3.00
Mix-ins & Toppings
Candied Jalapeño – 0.25
Make your chocolate a "hot" chocolate with these fresh-from-the-market peppers??
Pizelle Cone – 0.75
Homemade, hand rolled, and pretty too!
Almond Toffee – 0.50
Cocoa Nib Praline – 0.50
Cocoa nibs are simply roasted cocoa beans separated from the husks and broken into small bits. We take these beans and crystalize to create


9:36

Keep it Square

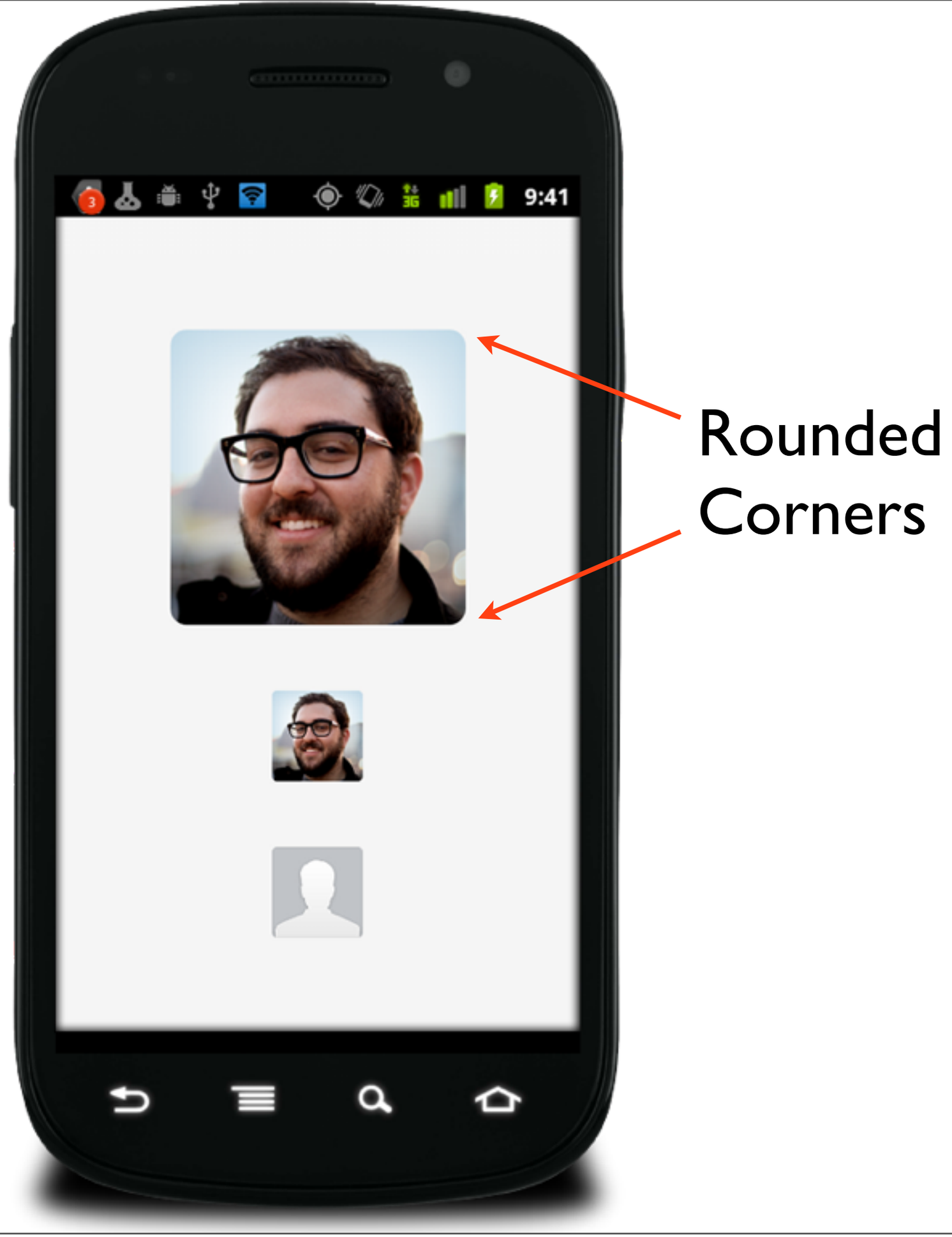
```
public class EditablePhoto extends View {

  @Override protected void onMeasure(int widthMeasureSpec,
      int heightMeasureSpec) {
    int measuredWidth = getDefaultSize(
        getSuggestedMinimumWidth(), widthMeasureSpec);
    int measuredHeight = getDefaultSize(
        getSuggestedMinimumHeight(), heightMeasureSpec);

    // Ensure this view is always square.
    int min = Math.min(measuredHeight, measuredWidth);
    setMeasuredDimension(min, min);
  }
}
```

Keep it Square

```
public class EditablePhoto extends View {

  @Override protected void onMeasure(int widthMeasureSpec,
      int heightMeasureSpec) {
    int measuredWidth = getDefaultSize(
        getSuggestedMinimumWidth(), widthMeasureSpec);
    int measuredHeight = getDefaultSize(
        getSuggestedMinimumHeight(), heightMeasureSpec);

    // Ensure this view is always square.
    int min = Math.min(measuredHeight, measuredWidth);
    setMeasuredDimension(min, min);
  }
}
```

Keep it Square

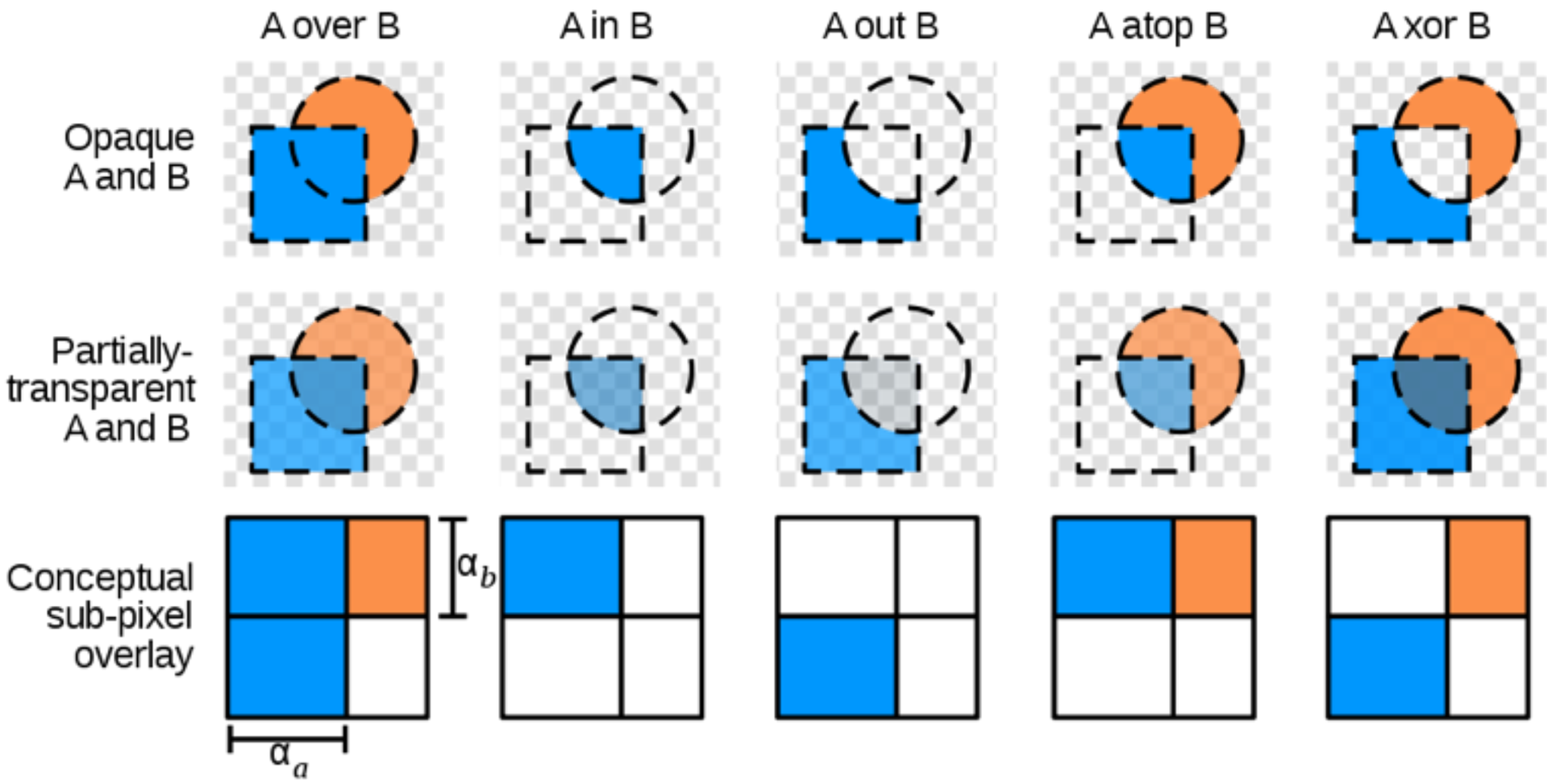
```
public class EditablePhoto extends View {

  @Override protected void onMeasure(int widthMeasureSpec,
      int heightMeasureSpec) {
    int measuredWidth = getDefaultSize(
        getSuggestedMinimumWidth(), widthMeasureSpec);
    int measuredHeight = getDefaultSize(
        getSuggestedMinimumHeight(), heightMeasureSpec);

    // Ensure this view is always square.
    int min = Math.min(measuredHeight, measuredWidth);
    setMeasuredDimension(min, min);
  }
}
```


9:41
Rounded
Corners

Don't Do This.

```
// Easy, but jagged corners.
Path roundRect = new Path();
path.addRoundRect(...);
canvas.clipPath(roundRect);
```

Alpha Compositing



Source: http://en.wikipedia.org/wiki/Alpha_compositing

```
paint.setXfermode(new PorterDuffXfermode(
    PorterDuff.Mode.SRC_IN));
```

Transparent Corners

```
Drawable imageDrawable = (image != null)
  ? new BitmapDrawable(image) : placeholder;

// An offscreen bitmap ensures antialiasing.
Bitmap output = Bitmap.createBitmap(size, size,
    Bitmap.Config.ARGB_8888);
Canvas canvas = new Canvas(output);

RectF outerRect = new RectF(0, 0, size, size);
float outerRadius = size / 18f;

// Draw a round rectangle in a solid color.
Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
paint.setColor(Color.RED);
canvas.drawRoundRect(outerRect, outerRadius,
    outerRadius, paint);
```

```
Drawable imageDrawable = (image != null)
  ? new BitmapDrawable(image) : placeholder;

// An offscreen bitmap ensures antialiasing.
Bitmap output = Bitmap.createBitmap(size, size,
    Bitmap.Config.ARGB_8888);
Canvas canvas = new Canvas(output);

RectF outerRect = new RectF(0, 0, size, size);
float outerRadius = size / 18f;

// Draw a round rectangle in a solid color.
Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
paint.setColor(Color.RED);
canvas.drawRoundRect(outerRect, outerRadius,
    outerRadius, paint);
```

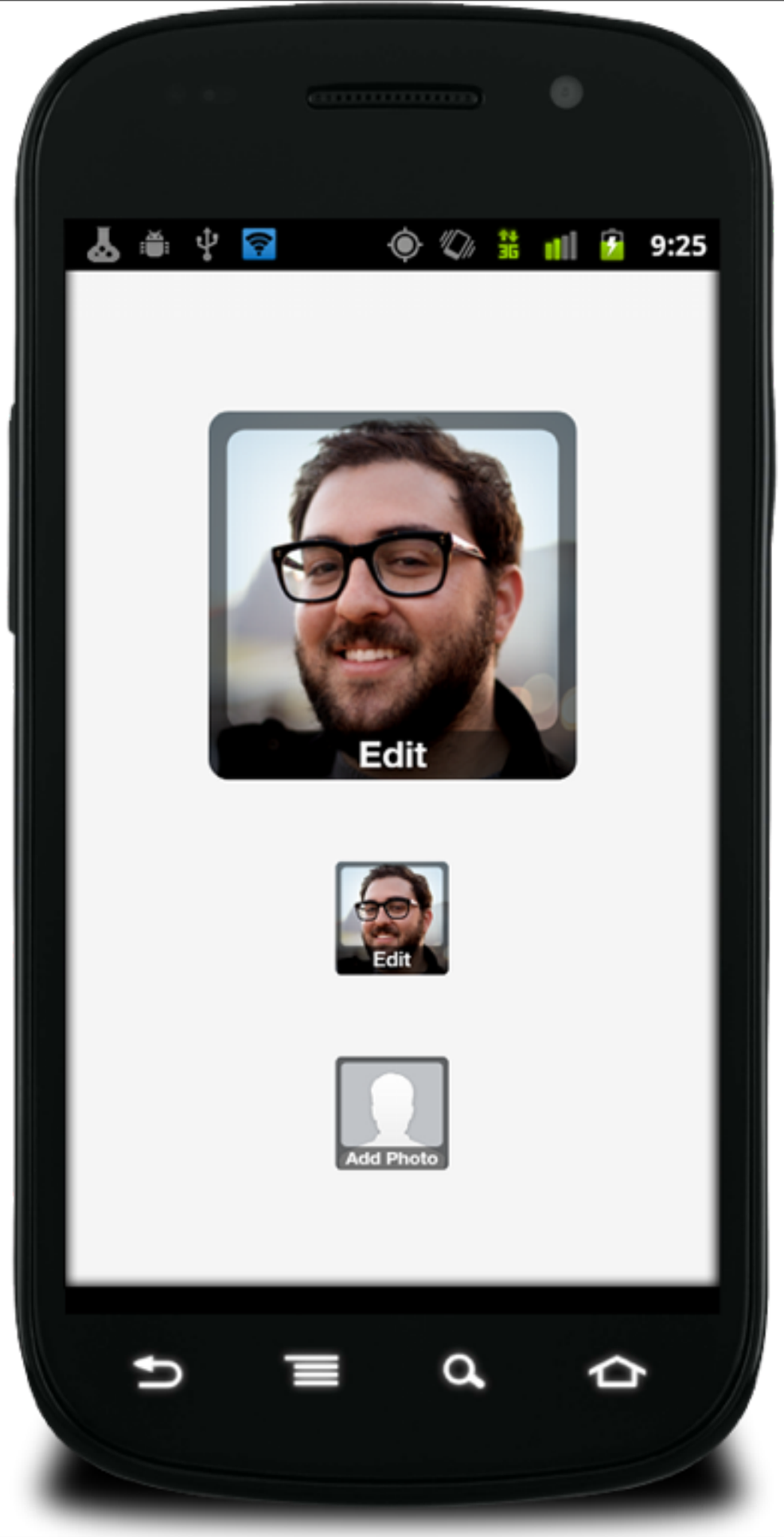
```
Drawable imageDrawable = (image != null)
  ? new BitmapDrawable(image) : placeholder;

// An offscreen bitmap ensures antialiasing.
Bitmap output = Bitmap.createBitmap(size, size,
    Bitmap.Config.ARGB_8888);
Canvas canvas = new Canvas(output);

RectF outerRect = new RectF(0, 0, size, size);
float outerRadius = size / 18f;

// Draw a round rectangle in a solid color.
Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
paint.setColor(Color.RED);
canvas.drawRoundRect(outerRect, outerRadius,
    outerRadius, paint);
```

```
Drawable imageDrawable = (image != null)
  ? new BitmapDrawable(image) : placeholder;

// An offscreen bitmap ensures antialiasing.
Bitmap output = Bitmap.createBitmap(size, size,
    Bitmap.Config.ARGB_8888);
Canvas canvas = new Canvas(output);

RectF outerRect = new RectF(0, 0, size, size);
float outerRadius = size / 18f;

// Draw a round rectangle in a solid color.
Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
paint.setColor(Color.RED);
canvas.drawRoundRect(outerRect, outerRadius,
    outerRadius, paint);
```

```
// Now compose the image with the round rectangle.
paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));

// Save the layer in order to apply the custom Paint when
// drawing the Drawable.
canvas.saveLayer(outerRect, paint, Paint.FILTER_BITMAP_FLAG);
imageDrawable.setBounds(0, 0, size, size);
imageDrawable.draw(canvas);
canvas.restore();
```

```
// Now compose the image with the round rectangle.
paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));

// Save the layer in order to apply the custom Paint when
// drawing the Drawable.
canvas.saveLayer(outerRect, paint, Paint.FILTER_BITMAP_FLAG);
imageDrawable.setBounds(0, 0, size, size);
imageDrawable.draw(canvas);
canvas.restore();
```

```
// Now compose the image with the round rectangle.
paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));

// Save the layer in order to apply the custom Paint when
// drawing the Drawable.
canvas.saveLayer(outerRect, paint, Paint.FILTER_BITMAP_FLAG);
imageDrawable.setBounds(0, 0, size, size);
imageDrawable.draw(canvas);
canvas.restore();
```


9:25
Edit
Edit
Add Photo

```
private Bitmap createEditFrame(int size, RectF outerRect,
    float outerRadius, float bottomThickness) {

  Bitmap output = Bitmap.createBitmap(size, size,
      Bitmap.Config.ARGB_8888);
  Canvas canvas = new Canvas(output);

  float thickness = size / 20f;

  RectF innerRect = new RectF(thickness, thickness,
      size - thickness, size - bottomThickness);
  float innerRadius = size / 20f;
  Paint innerPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
  innerPaint.setColor(Color.RED);
```

```
private Bitmap createEditFrame(int size, RectF outerRect,
    float outerRadius, float bottomThickness) {

  Bitmap output = Bitmap.createBitmap(size, size,
      Bitmap.Config.ARGB_8888);
  Canvas canvas = new Canvas(output);

  float thickness = size / 20f;

  RectF innerRect = new RectF(thickness, thickness,
      size - thickness, size - bottomThickness);
  float innerRadius = size / 20f;
  Paint innerPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
  innerPaint.setColor(Color.RED);
```

```
private Bitmap createEditFrame(int size, RectF outerRect,
    float outerRadius, float bottomThickness) {

  Bitmap output = Bitmap.createBitmap(size, size,
      Bitmap.Config.ARGB_8888);
  Canvas canvas = new Canvas(output);

  float thickness = size / 20f;

  RectF innerRect = new RectF(thickness, thickness,
      size - thickness, size - bottomThickness);
  float innerRadius = size / 20f;
  Paint innerPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
  innerPaint.setColor(Color.RED);
```

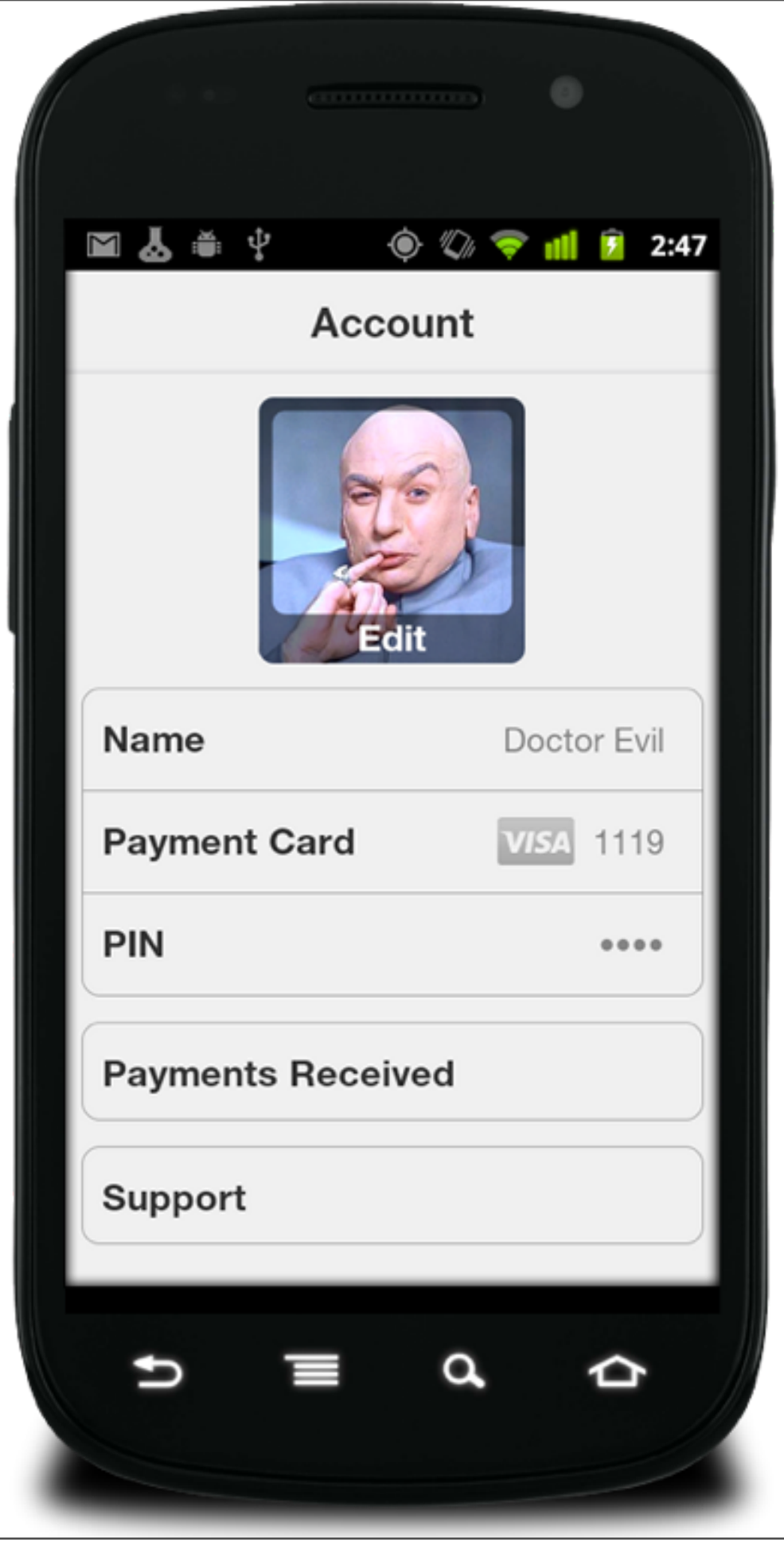
```
private Bitmap createEditFrame(int size, RectF outerRect,
    float outerRadius, float bottomThickness) {

  Bitmap output = Bitmap.createBitmap(size, size,
      Bitmap.Config.ARGB_8888);
  Canvas canvas = new Canvas(output);

  float thickness = size / 20f;

  RectF innerRect = new RectF(thickness, thickness,
      size - thickness, size - bottomThickness);
  float innerRadius = size / 20f;
  Paint innerPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
  innerPaint.setColor(Color.RED);
```

```
// Draw the inner viewport in a solid color.
canvas.drawRoundRect(innerRect, innerRadius,
    innerRadius, innerPaint);

// Now compose a larger rounded rectangle. The inner viewport
// will be punched out.
Paint outerPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
outerPaint.setColor(getResources().getColor(
    R.color.editable_photo_border));

outerPaint.setXfermode(
    new PorterDuffXfermode(PorterDuff.Mode.SRC_OUT));

  canvas.drawRoundRect(outerRect, outerRadius,
      outerRadius, outerPaint);

  return output;
}
```

```
    // Draw the inner viewport in a solid color.
    canvas.drawRoundRect(innerRect, innerRadius,
            innerRadius, innerPaint);

    // Now compose a larger rounded rectangle. The inner viewport
    // will be punched out.
    Paint outerPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    outerPaint.setColor(getResources().getColor(
            R.color.editable_photo_border));

    outerPaint.setXfermode(
            new PorterDuffXfermode(PorterDuff.Mode.SRC_OUT));

    canvas.drawRoundRect(outerRect, outerRadius,
            outerRadius, outerPaint);

    return output;
}
```

```
        // Draw the inner viewport in a solid color.
        canvas.drawRoundRect(innerRect, innerRadius,
                innerRadius, innerPaint);

        // Now compose a larger rounded rectangle. The inner viewport
        // will be punched out.
        Paint outerPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        outerPaint.setColor(getResources().getColor(
                R.color.editable_photo_border));

        outerPaint.setXfermode(
                new PorterDuffXfermode(PorterDuff.Mode.SRC_OUT));

        canvas.drawRoundRect(outerRect, outerRadius,
                outerRadius, outerPaint);

        return output;
    }
```

```
        // Draw the inner viewport in a solid color.
        canvas.drawRoundRect(innerRect, innerRadius,
                innerRadius, innerPaint);

        // Now compose a larger rounded rectangle. The inner viewport
        // will be punched out.
        Paint outerPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        outerPaint.setColor(getResources().getColor(
                R.color.editable_photo_border));

        outerPaint.setXfermode(
                new PorterDuffXfermode(PorterDuff.Mode.SRC_OUT));

        canvas.drawRoundRect(outerRect, outerRadius,
                outerRadius, outerPaint);

        return output;
    }
```


2:47
Account
Edit
Name
Doctor Evil
Payment Card
VISA 1119
PIN
Payments Received
Support


2:50
Account
Edit
Name
Doctor Evil
Payment Card
VISA 1119
PIN
Payments Received
Support

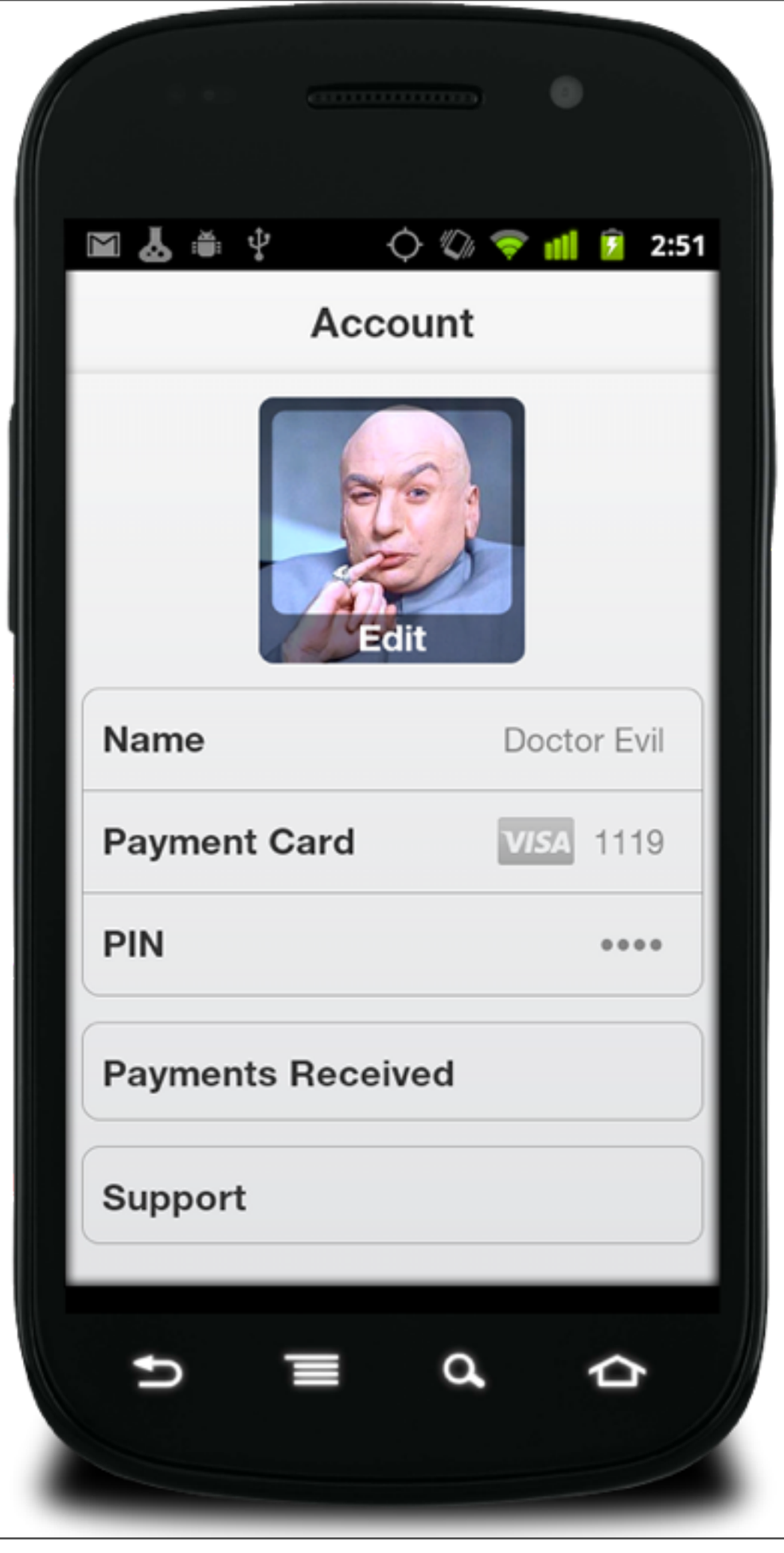
plastic_background.xml

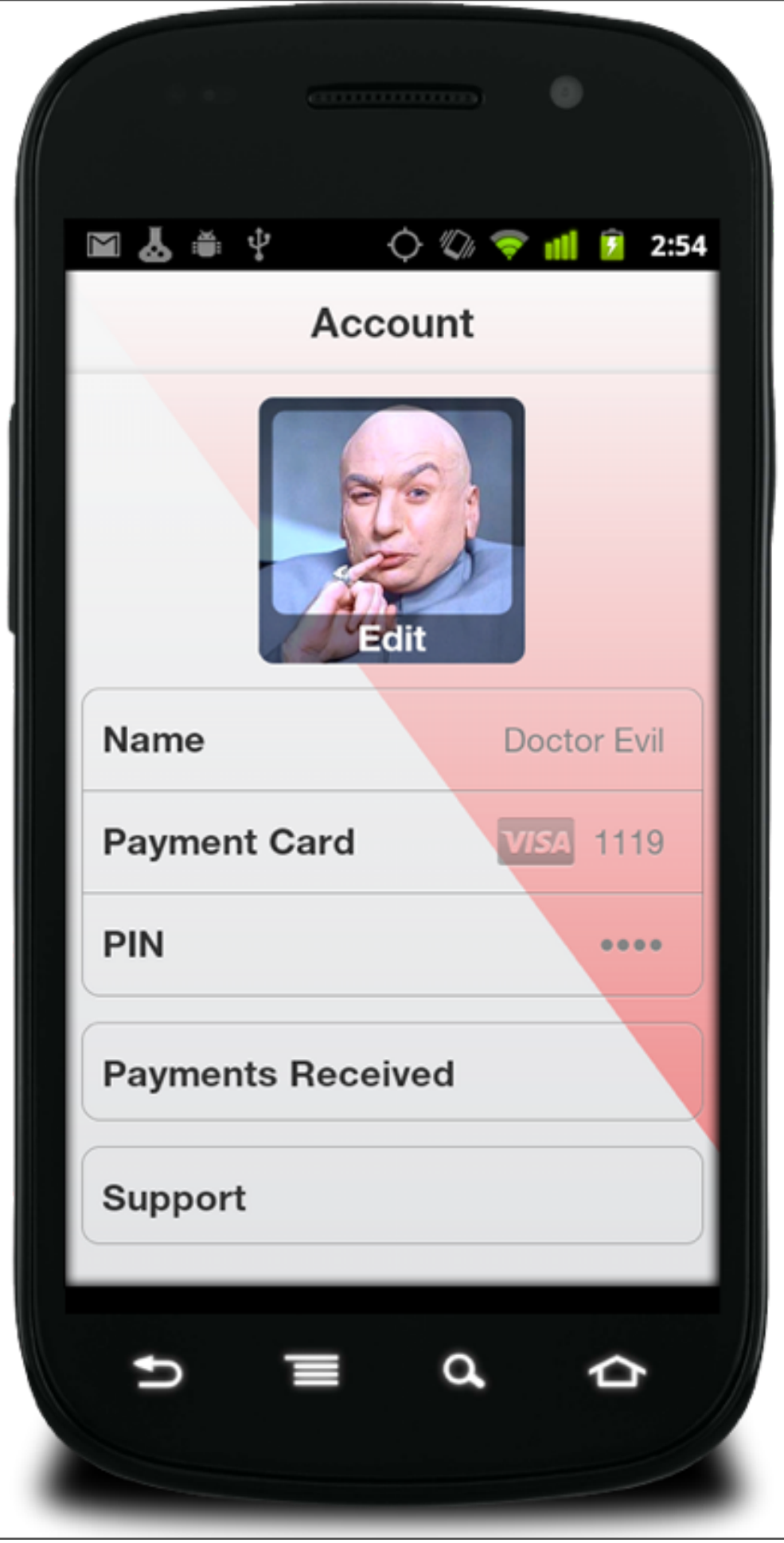
res/drawable/plastic_background.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">
  <!-- Nearly white at the top, light gray at the bottom. -->
  <gradient
    android:startColor="#fff7f7f7"
    android:endColor="#ffe2e3e5"
    android:angle="270"/>
</shape>
```

PlasticLinearLayout

```
public class PlasticLinearLayout extends LinearLayout {

  public PlasticLinearLayout(Context context, AttributeSet attrs) {
    super(context, attrs);

    // The subtle gradient draws behind everything.
    setBackgroundResource(R.drawable.plastic_window_background);
  }
}
```


2:51
Account
Edit
Name
Doctor Evil
Payment Card
VISA 1119
PIN
Payments Received
Support


2:54
Account
Edit
Name
Doctor Evil
Payment Card
VISA 1119
PIN
••••
Payments Received
Support

Triangular Shine

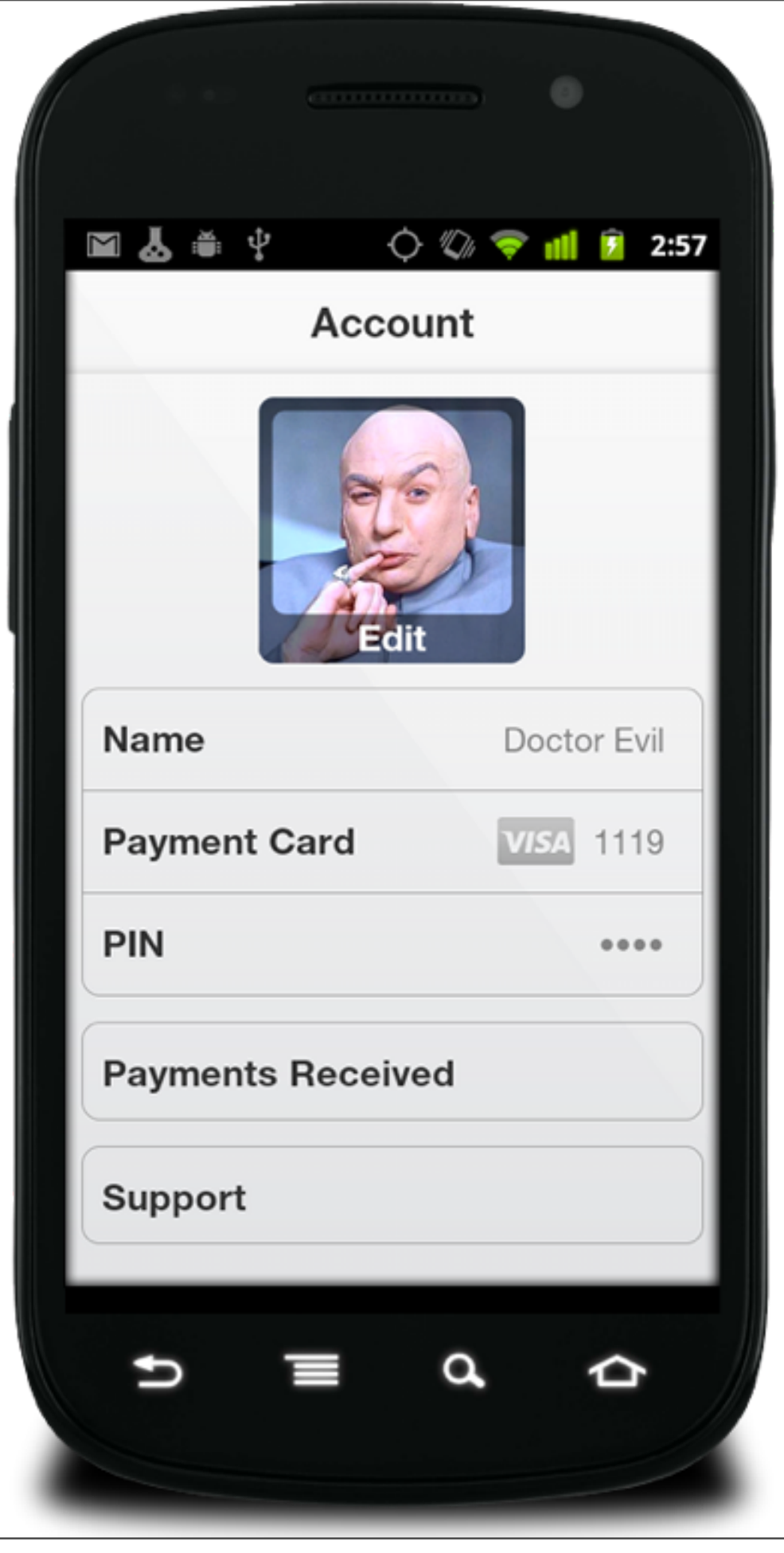
```
private void createShinePath() {
  int width = getWidth();
  int height = (int) (0.85 * getHeight());

  shinePath = new Path();
  shinePath.moveTo(0, 0);
  shinePath.lineTo(width, 0);
  shinePath.lineTo(width, height);
  shinePath.close();

  shinePaint.setShader(new LinearGradient(0, 0, 0, height,
      0x66ffffff, 0x10ffffff, CLAMP));
}
```

Triangular Shine

```
private void createShinePath() {
  int width = getWidth();
  int height = (int) (0.85 * getHeight());

  shinePath = new Path();
  shinePath.moveTo(0, 0);
  shinePath.lineTo(width, 0);
  shinePath.lineTo(width, height);
  shinePath.close();

  shinePaint.setShader(new LinearGradient(0, 0, 0, height,
      0x66ffffff, 0x10ffffff, CLAMP));
}
```

Triangular Shine

```
private void createShinePath() {
  int width = getWidth();
  int height = (int) (0.85 * getHeight());

  shinePath = new Path();
  shinePath.moveTo(0, 0);
  shinePath.lineTo(width, 0);
  shinePath.lineTo(width, height);
  shinePath.close();

  shinePaint.setShader(new LinearGradient(0, 0, 0, height,
      0x66ffffff, 0x10ffffff, CLAMP));
}
```


2:57
Account
Edit
Name
Doctor Evil
Payment Card
VISA 1119
PIN
Payments Received
Support

```
public class PlasticLinearLayout extends LinearLayout {
  private final Paint shinePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
  private Path shinePath;

  public PlasticLinearLayout(Context context, AttributeSet attrs) {
    super(context, attrs);

    // The subtle gradient draws behind everything.
    setBackgroundResource(R.drawable.plastic_background);
  }

  @Override protected void dispatchDraw(Canvas canvas) {
    if (shinePath == null) {
      createShinePath();
    }
    // Draw the shine behind the children.
    canvas.drawPath(shinePath, shinePaint);

    // Draw the children.
    super.dispatchDraw(canvas);
  }
```

```
public class PlasticLinearLayout extends LinearLayout {
  private final Paint shinePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
  private Path shinePath;

  public PlasticLinearLayout(Context context, AttributeSet attrs) {
    super(context, attrs);

    // The subtle gradient draws behind everything.
    setBackgroundResource(R.drawable.plastic_background);
  }

  @Override protected void dispatchDraw(Canvas canvas) {
    if (shinePath == null) {
      createShinePath();
    }
    // Draw the shine behind the children.
    canvas.drawPath(shinePath, shinePaint);

    // Draw the children.
    super.dispatchDraw(canvas);
  }
```

```
public class PlasticLinearLayout extends LinearLayout {
  private final Paint shinePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
  private Path shinePath;

  public PlasticLinearLayout(Context context, AttributeSet attrs) {
    super(context, attrs);

    // The subtle gradient draws behind everything.
    setBackgroundResource(R.drawable.plastic_background);
  }

  @Override protected void dispatchDraw(Canvas canvas) {
    if (shinePath == null) {
      createShinePath();
    }
    // Draw the shine behind the children.
    canvas.drawPath(shinePath, shinePaint);

    // Draw the children.
    super.dispatchDraw(canvas);
  }
```

```
public class PlasticLinearLayout extends LinearLayout {
  private final Paint shinePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
  private Path shinePath;

  public PlasticLinearLayout(Context context, AttributeSet attrs) {
    super(context, attrs);

    // The subtle gradient draws behind everything.
    setBackgroundResource(R.drawable.plastic_background);
  }

  @Override protected void dispatchDraw(Canvas canvas) {
    if (shinePath == null) {
      createShinePath();
    }
    // Draw the shine behind the children.
    canvas.drawPath(shinePath, shinePaint);

    // Draw the children.
    super.dispatchDraw(canvas);
  }
```

```
public class PlasticLinearLayout extends LinearLayout {
  private final Paint shinePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
  private Path shinePath;

  public PlasticLinearLayout(Context context, AttributeSet attrs) {
    super(context, attrs);

    // The subtle gradient draws behind everything.
    setBackgroundResource(R.drawable.plastic_background);
  }

  @Override protected void dispatchDraw(Canvas canvas) {
    if (shinePath == null) {
      createShinePath();
    }
    // Draw the shine behind the children.
    canvas.drawPath(shinePath, shinePaint);

    // Draw the children.
    super.dispatchDraw(canvas);
  }
```

```
public class PlasticLinearLayout extends LinearLayout {
  private final Paint shinePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
  private Path shinePath;

  public PlasticLinearLayout(Context context, AttributeSet attrs) {
    super(context, attrs);

    // The subtle gradient draws behind everything.
    setBackgroundResource(R.drawable.plastic_background);
  }

  @Override protected void dispatchDraw(Canvas canvas) {
    if (shinePath == null) {
      createShinePath();
    }
    // Draw the shine behind the children.
    canvas.drawPath(shinePath, shinePaint);

    // Draw the children.
    super.dispatchDraw(canvas);
  }
```

```
@Override protected void onLayout(boolean changed,
    int l, int t, int r, int b) {
  super.onLayout(changed, l, t, r, b);

  // Invalidate the path whenever the size changes.
  shinePath = null;
}

/**
 * Creates a triangle shape that draws the subtle glossy
 * shine.
 */
  private void createShinePath() {
    ...
  }
}
```

```
@Override protected void onLayout(boolean changed,
    int l, int t, int r, int b) {
  super.onLayout(changed, l, t, r, b);

  // Invalidate the path whenever the size changes.
  shinePath = null;
}

/**
 * Creates a triangle shape that draws the subtle glossy
 * shine.
 */
  private void createShinePath() {
    ...
  }
}
```

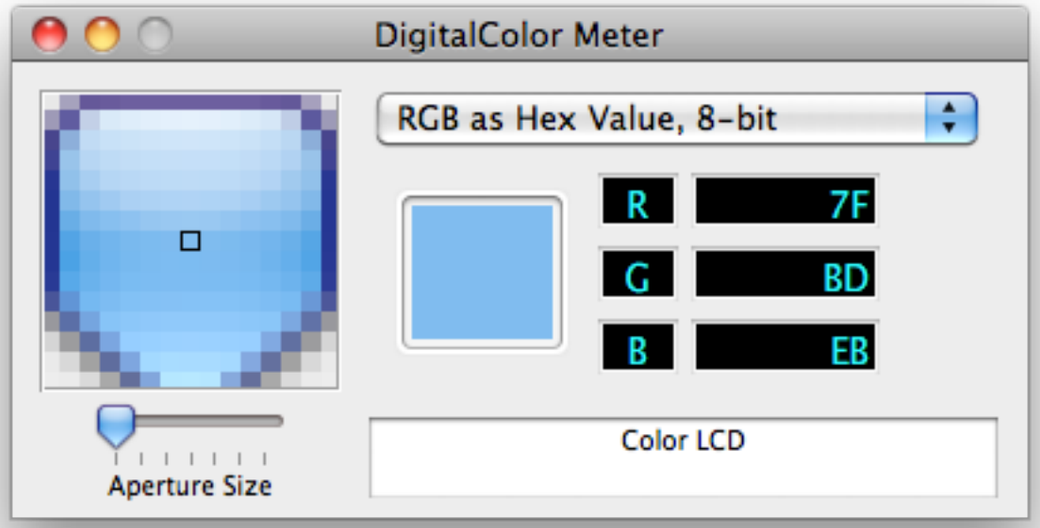
Usage

```
<com.squareup.ui.PlasticLinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    >

  ...normal layout here

</com.squareup.ui.PlasticLinearLayout>
```

Takeaways

- Subtle gradients FTW
- Test with bright, primary colors
- Invalidate drawables when bounds change

Pro Tip: DigitalColor Meter



Custom Attributes

res/values/attrs.xml

```
<declare-styleable name="EditablePhoto">
  <attr name="caption" format="string"/>
  <attr name="captionSize" format="dimension"/>
  <attr name="placeholder" format="reference"/>
</declare-styleable>
```

Attribute Namespace

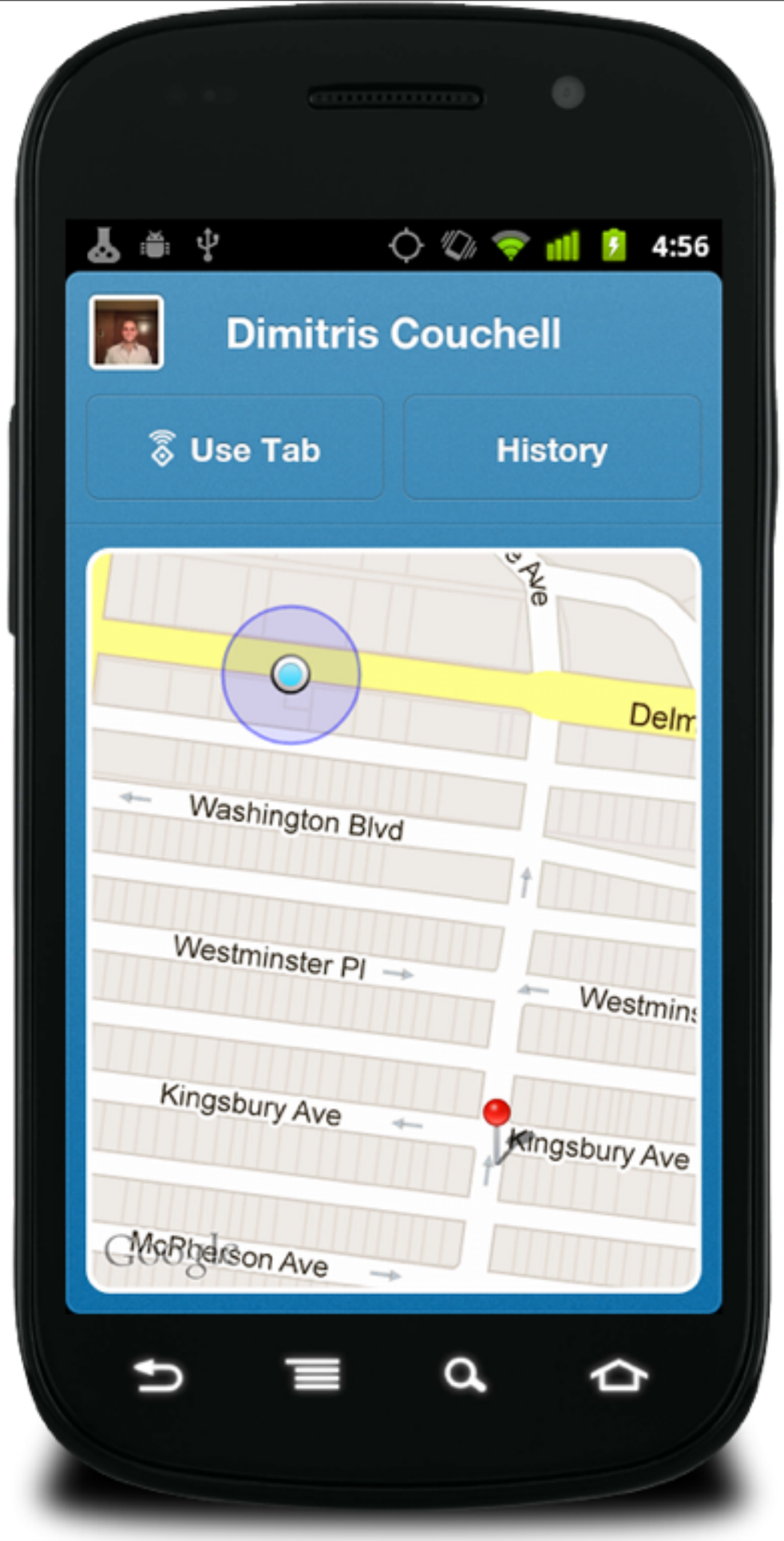
```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:square="http://schemas.android.com/apk/res/com.squareup"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:gravity="center_horizontal|top"
  >
```

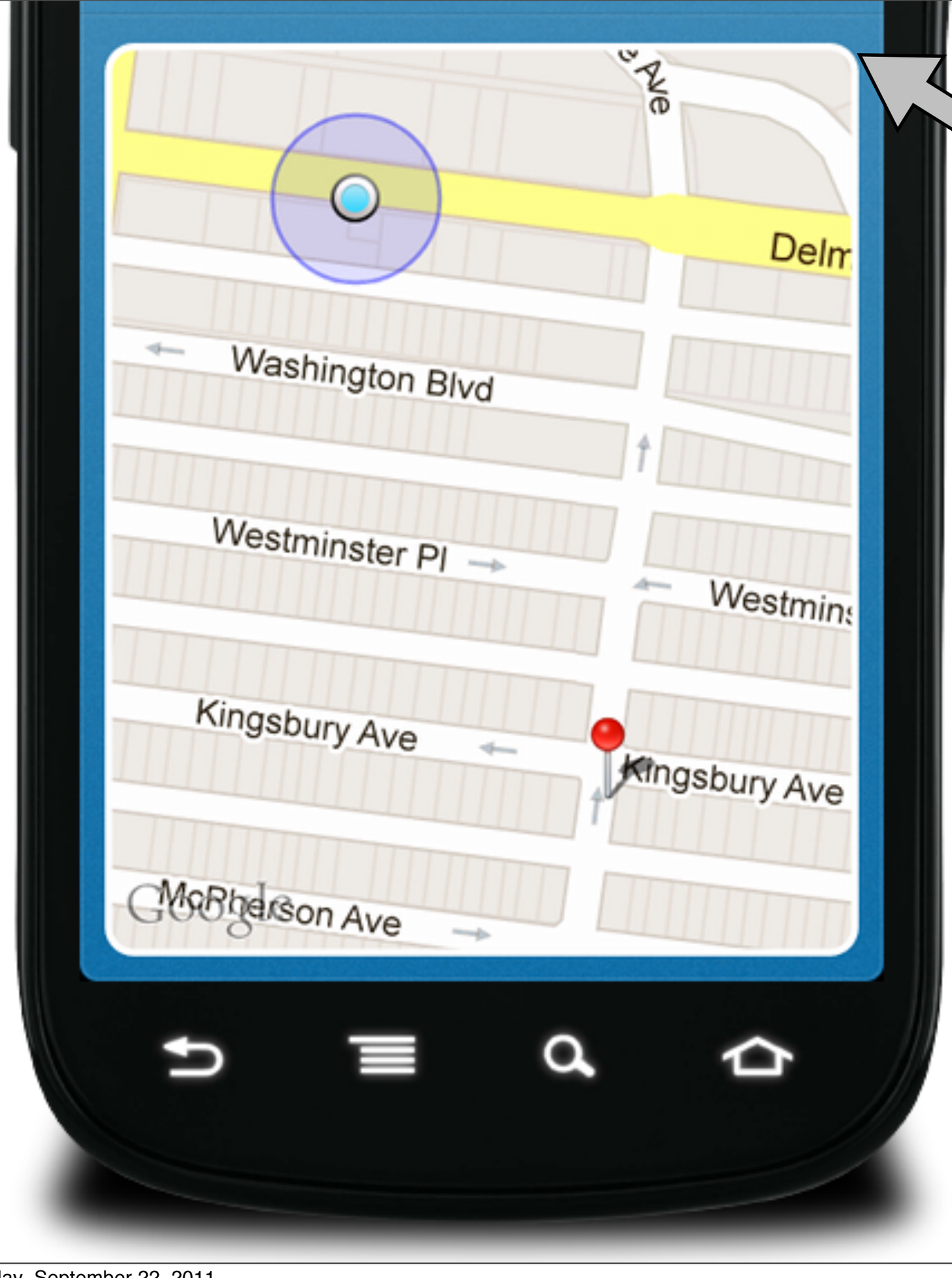
Your Application Package

Referencing Attributes

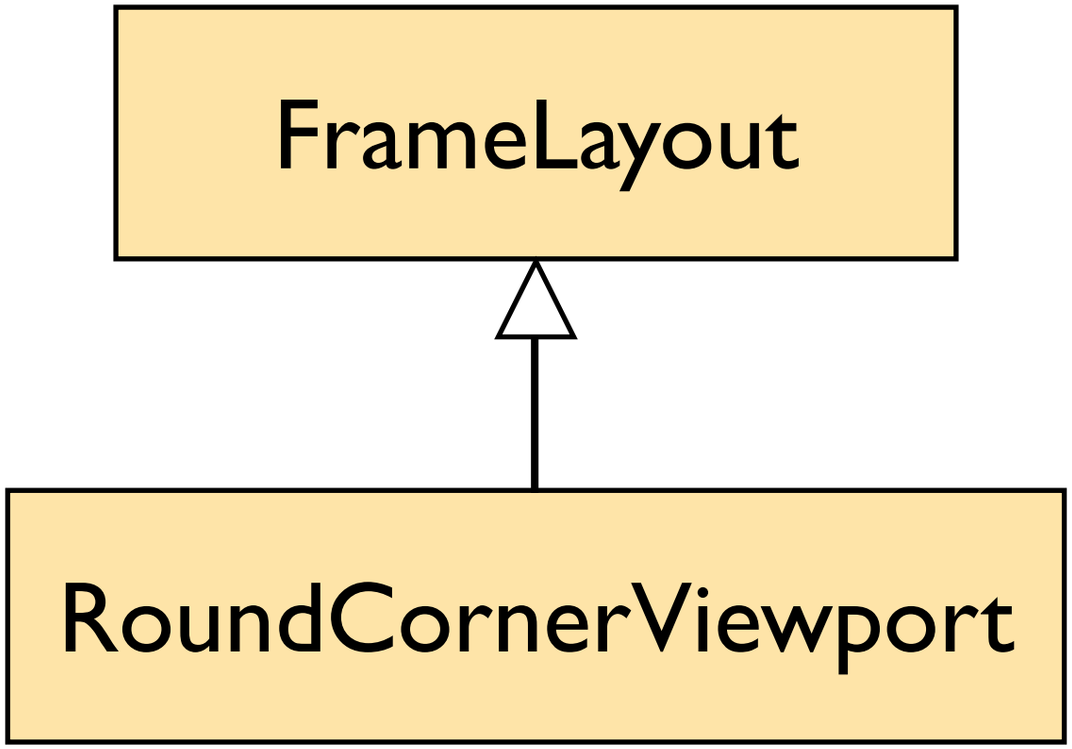
```
<com.squareup.EditablePhoto
    android:layout_width="@dimen/merchant_photo"
    android:layout_height="@dimen/merchant_photo"
    square:caption="@string/edit"
    square:captionSize="@dimen/merchant_caption"
    square:placeholder="@drawable/merchant_placeholder"
    />
```

```
public EditablePhoto(Context context, AttributeSet attrs) {
  super(context, attrs);

  TypedArray a = context.obtainStyledAttributes(attrs,
      R.styleable.EditablePhoto, 0, 0);

  placeholder = a.getDrawable(
      R.styleable.EditablePhoto_placeholder);
  caption = a.getString(
      R.styleable.EditablePhoto_caption);
  captionSize = a.getDimension(
      R.styleable.EditablePhoto_captionSize,
          getResources().getDimension(
              R.dimen.editable_photo_default_caption_size));

  a.recycle();
  ...
}
```


4:56
Dimitris Couchell
Use Tab
History
Washington Blvd
Westminster Pl
Kingsbury Ave
Kingsbury Ave
McPherson Ave



Round Corners

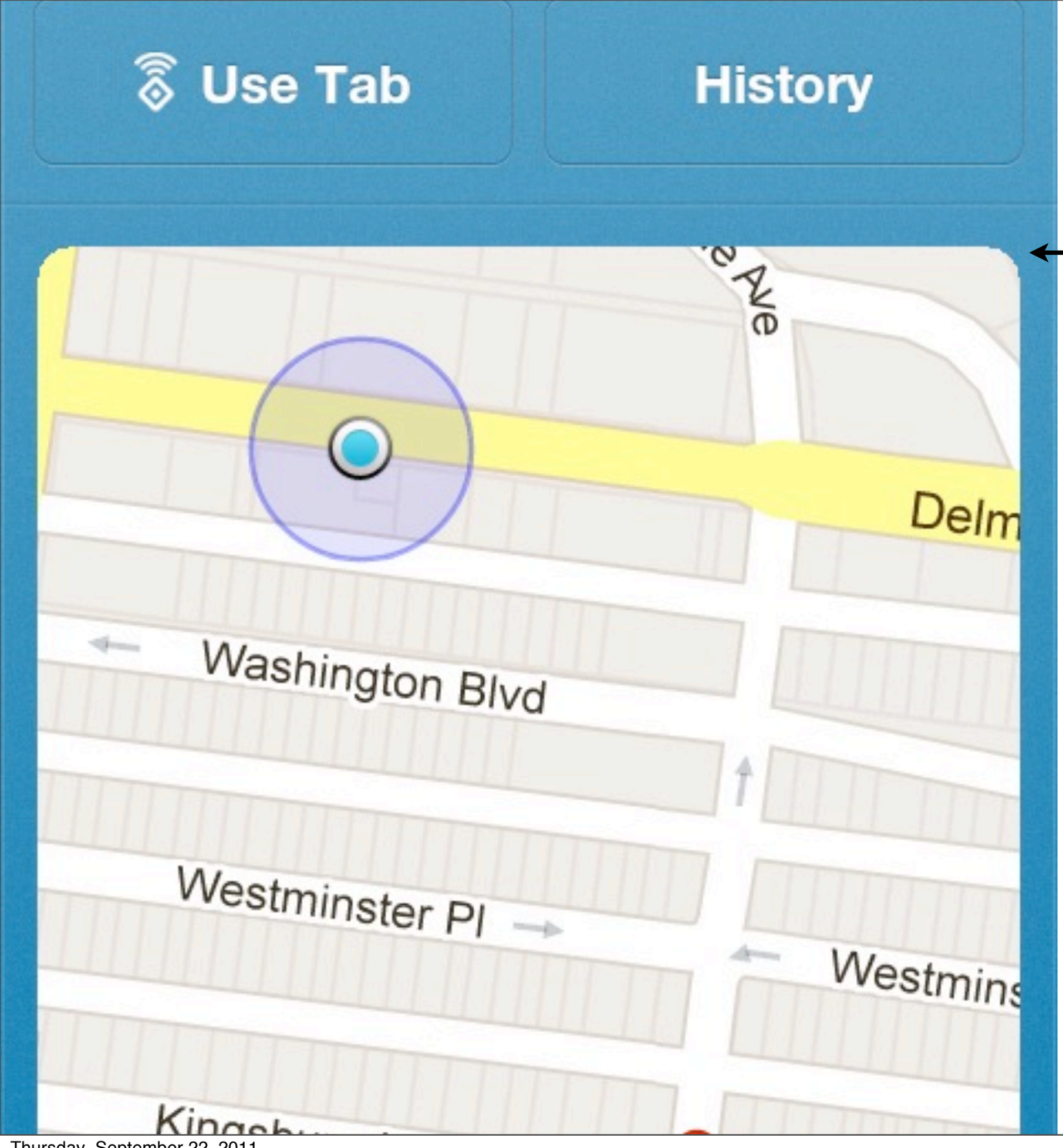

FrameLayout
RoundCornerViewport

```
public class RoundCornerViewport extends FrameLayout {
  @Override protected void dispatchDraw(Canvas canvas) {
    // Clip to a round rect shape. This will leave some
    // jagged pixels around the rounded corners.
    canvas.save(Canvas.CLIP_SAVE_FLAG);
    canvas.clipPath(roundRectPath);

    // Draw the child view, but only the part inside the
    // clip path shows.
    super.dispatchDraw(canvas);

    canvas.restore();

    // Cover up the jagged pixels with this border.
    canvas.drawRoundRect(boundsRect, cornerRadius,
        cornerRadius, borderLinePaint);
  }
}
```

```
public class RoundCornerViewport extends FrameLayout {
  @Override protected void dispatchDraw(Canvas canvas) {
    // Clip to a round rect shape. This will leave some
    // jagged pixels around the rounded corners.
    canvas.save(Canvas.CLIP_SAVE_FLAG);
    canvas.clipPath(roundRectPath);

    // Draw the child view, but only the part inside the
    // clip path shows.
    super.dispatchDraw(canvas);

    canvas.restore();

    // Cover up the jagged pixels with this border.
    canvas.drawRoundRect(boundsRect, cornerRadius,
        cornerRadius, borderLinePaint);
  }
}
```

```
public class RoundCornerViewport extends FrameLayout {
  @Override protected void dispatchDraw(Canvas canvas) {
    // Clip to a round rect shape. This will leave some
    // jagged pixels around the rounded corners.
    canvas.save(Canvas.CLIP_SAVE_FLAG);
    canvas.clipPath(roundRectPath);

    // Draw the child view, but only the part inside the
    // clip path shows.
    super.dispatchDraw(canvas);

    canvas.restore();

    // Cover up the jagged pixels with this border.
    canvas.drawRoundRect(boundsRect, cornerRadius,
        cornerRadius, borderLinePaint);
  }
}
```

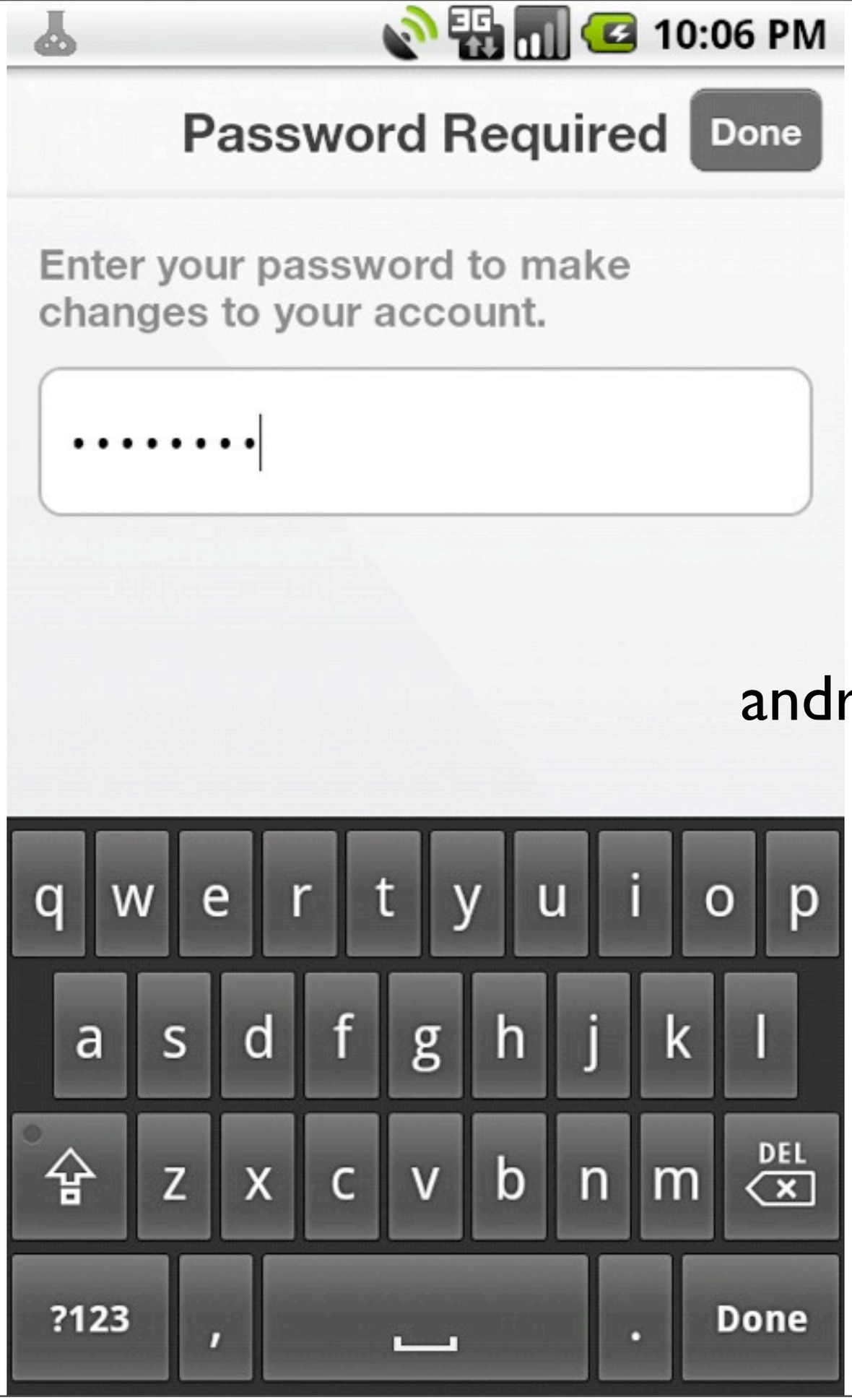
```
public class RoundCornerViewport extends FrameLayout {
  @Override protected void dispatchDraw(Canvas canvas) {
    // Clip to a round rect shape. This will leave some
    // jagged pixels around the rounded corners.
    canvas.save(Canvas.CLIP_SAVE_FLAG);
    canvas.clipPath(roundRectPath);

    // Draw the child view, but only the part inside the
    // clip path shows.
    super.dispatchDraw(canvas);

    canvas.restore();

    // Cover up the jagged pixels with this border.
    canvas.drawRoundRect(boundsRect, cornerRadius,
        cornerRadius, borderLinePaint);
  }
}
```



Jagged Corner.

```
public class RoundCornerViewport extends FrameLayout {
  @Override protected void dispatchDraw(Canvas canvas) {
    // Clip to a round rect shape. This will leave some
    // jagged pixels around the rounded corners.
    canvas.save(Canvas.CLIP_SAVE_FLAG);
    canvas.clipPath(roundRectPath);

    // Draw the child view, but only the part inside the
    // clip path shows.
    super.dispatchDraw(canvas);

    canvas.restore();

    // Cover up the jagged pixels with this border.
    canvas.drawRoundRect(boundsRect, cornerRadius,
        cornerRadius, borderLinePaint);
  }
}
```

Thick Line



android:imeOptions="actionDone"

```
passwordField.setOnEditorActionListener(
    new TextView.OnEditorActionListener() {
      @Override public boolean onEditorAction(
          TextView v, int actionId, KeyEvent event) {

        if (actionId == EditorInfo.IME_ACTION_DONE) {
          doneClicked(passwordField);
          return true;
        }

        return false;
      }
    });
```

```
passwordField.setOnEditorActionListener(
    new TextView.OnEditorActionListener() {
      @Override public boolean onEditorAction(
          TextView v, int actionId, KeyEvent event) {

        if (actionId == EditorInfo.IME_ACTION_DONE) {
          doneClicked(passwordField);
          return true;
        }

        return false;
      }
    });
```

More to Explore

```
android:imeOptions=”actionNext”

android:inputType=”phone”

android:inputType=”textPersonName|textCapWords”

etc...
```

Thinking of using ViewFlipper?

http://www.flickr.com/photos/cjsorg/2460253775/

Bug 6191

Fatal bug prior to Honeycomb:
http://code.google.com/p/android/issues/detail?id=6191

SafeViewFlipper

```
@Override protected void onDetachedFromWindow() {
    try {
        super.onDetachedFromWindow();
    } catch (IllegalArgumentException e) {
        stopFlipping();
    }
}
```

http://pastie.org/1086467


MOTOROLA
10:03 AM
M.D
Maja Daley
Sightglass Coffee
Toasty Melts LLC
Smitten Ice Cream
The Rib Whip
Explore Places

getChildDrawingOrder(...)

drawChild(...)

← canvas.clipRect(...)

canvas.translate(...)
canvas.rotate(...)

Holograms

Start with three images:

Hologram Interface

```
public interface Hologram {

  /**
   * Adjusts the hologram display.
   *
   * @param value ranges from -1..1.
   */
  void update(float value);
}
```

```
public class HologramView extends View implements Hologram {
  private final Bitmap baseImage;
  private final Bitmap redOverlay;
  private final Bitmap blueOverlay;
  private final Paint overlayPaint = new Paint();
  private float value;

  ...

  @Override public void draw(Canvas canvas) {
    canvas.drawBitmap(baseImage, 0, 0, null);

    Bitmap overlay = (value >= 0) ? redOverlay : blueOverlay;

    int overlayAlpha = Math.abs((int) 255 * value);

    overlayPaint.setAlpha(overlayAlpha);
    canvas.drawBitmap(overlay, 0, 0, overlayPaint);
  }

  @Override public void update(float value) {
    this.value = value;
    invalidate();
  }
}
```

```
public class HologramView extends View implements Hologram {
  private final Bitmap baseImage;
  private final Bitmap redOverlay;
  private final Bitmap blueOverlay;
  private final Paint overlayPaint = new Paint();
  private float value;

  ...

  @Override public void draw(Canvas canvas) {
    canvas.drawBitmap(baseImage, 0, 0, null);

    Bitmap overlay = (value >= 0) ? redOverlay : blueOverlay;

    int overlayAlpha = Math.abs((int) 255 * value);

    overlayPaint.setAlpha(overlayAlpha);
    canvas.drawBitmap(overlay, 0, 0, overlayPaint);
  }

  @Override public void update(float value) {
    this.value = value;
    invalidate();
  }
}
```

```
public class HologramView extends View implements Hologram {
  private final Bitmap baseImage;
  private final Bitmap redOverlay;
  private final Bitmap blueOverlay;
  private final Paint overlayPaint = new Paint();
  private float value;

  ...

  @Override public void draw(Canvas canvas) {
    canvas.drawBitmap(baseImage, 0, 0, null);

    Bitmap overlay = (value >= 0) ? redOverlay : blueOverlay;

    int overlayAlpha = Math.abs((int) 255 * value);

    overlayPaint.setAlpha(overlayAlpha);
    canvas.drawBitmap(overlay, 0, 0, overlayPaint);
  }

  @Override public void update(float value) {
    this.value = value;
    invalidate();
  }
}
```

```
public class HologramView extends View implements Hologram {
  private final Bitmap baseImage;
  private final Bitmap redOverlay;
  private final Bitmap blueOverlay;
  private final Paint overlayPaint = new Paint();
  private float value;

  ...

  @Override public void draw(Canvas canvas) {
    canvas.drawBitmap(baseImage, 0, 0, null);

    Bitmap overlay = (value >= 0) ? redOverlay : blueOverlay;

    int overlayAlpha = Math.abs((int) 255 * value);

    overlayPaint.setAlpha(overlayAlpha);
    canvas.drawBitmap(overlay, 0, 0, overlayPaint);
  }

  @Override public void update(float value) {
    this.value = value;
    invalidate();
  }
}
```

```
public class HologramView extends View implements Hologram {
  private final Bitmap baseImage;
  private final Bitmap redOverlay;
  private final Bitmap blueOverlay;
  private final Paint overlayPaint = new Paint();
  private float value;

  ...

  @Override public void draw(Canvas canvas) {
    canvas.drawBitmap(baseImage, 0, 0, null);

    Bitmap overlay = (value >= 0) ? redOverlay : blueOverlay;

    int overlayAlpha = Math.abs((int) 255 * value);

    overlayPaint.setAlpha(overlayAlpha);
    canvas.drawBitmap(overlay, 0, 0, overlayPaint);
  }

  @Override public void update(float value) {
    this.value = value;
    invalidate();
  }
}
```

```
public class HologramView extends View implements Hologram {
  private final Bitmap baseImage;
  private final Bitmap redOverlay;
  private final Bitmap blueOverlay;
  private final Paint overlayPaint = new Paint();
  private float value;

  ...

  @Override public void draw(Canvas canvas) {
    canvas.drawBitmap(baseImage, 0, 0, null);

    Bitmap overlay = (value >= 0) ? redOverlay : blueOverlay;

    int overlayAlpha = Math.abs((int) 255 * value);

    overlayPaint.setAlpha(overlayAlpha);
    canvas.drawBitmap(overlay, 0, 0, overlayPaint);
  }

  @Override public void update(float value) {
    this.value = value;
    invalidate();
  }
}
```

Accelerometer Listener

```
@Override public void onSensorChanged(SensorEvent sensorEvent) {
  final float ax = sensorEvent.values[0];
  final float ay = sensorEvent.values[1];
  final float az = sensorEvent.values[2];

  // Sum typically ranges from approximately -16...16
  float sum = ax + ay + az;

  // Dividing by 4 looks nice. Smaller denominators make the
  // hologram shift more quickly.
  float value = (float) Math.sin(sum / 4);
  delegate.update(value);
}
```

Boxcar Filter

- Averages the accelerometer readings
  - Contains an array, length = 8
- Smooths the otherwise jittery signal

Demo

Questions

eric@squareup.com