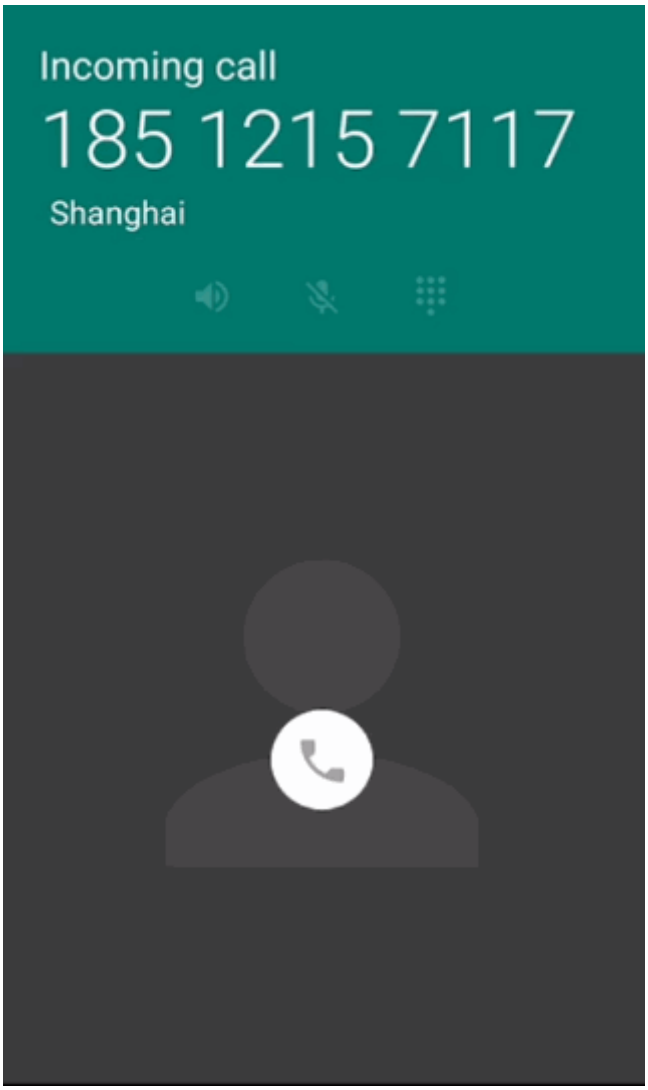
基于Android7.0的分析




```
<com.android.incallui.GlowPadWrapper
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:dc="http://schemas.android.com/apk/res-auto"
        android:id="@+id/glow_pad_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:focusable="true"
        android:layout_centerHorizontal="true"
        android:gravity="center"
        android:background="@color/glowpad_background_color"
        android:layout_marginBottom="@dimen/glowpadview_margin_bottom"

        dc:targetDrawables="@array/incoming_call_widget_audio_with_sms_targets"
        dc:targetDescriptions="@array/incoming_call_widget_audio_with_sms_target_descriptions"
        dc:directionDescriptions="@array/incoming_call_widget_audio_with_sms_direction_descriptions"
        dc:handleDrawable="@drawable/ic_incall_audio_handle"
        dc:outerRingDrawable="@drawable/ic_lockscreen_outerring"
        dc:outerRadius="@dimen/glowpadview_target_placement_radius"
        dc:innerRadius="@dimen/glowpadview_inner_radius"
        dc:snapMargin="@dimen/glowpadview_snap_margin"
        dc:feedbackCount="1"
        dc:vibrationDuration="20"
        dc:glowRadius="@dimen/glowpadview_glow_radius"
        dc:pointDrawable="@drawable/ic_lockscreen_glowdot"
        dc:allowScaling="true" />
```

自定义属性xml文件

（http://androidxref.com/7.0.0_r1/xref/packages/apps/Dialer/InCallUI/res/values/attrs.xml）

targetDrawables、handleDrawable 、outerRingDrawable、pointDrawable：自定义Drawable

targetDescriptions、directionDescriptions：同contentDescription的用途一样为残疾人士app服务

outerRadius、innerRadius：从内圈到外圈的范围是光点的区域，outerRadius表示外圈的半径，innerRadius表示内圈的半径

snapMargin：如果正在滑动的手到target的距离小于snapMargin，那么outerRingDrawable、targetDrawables的转台就会发生变化，图标就会显示出来。

glowRadius：

来自http://blog.csdn.net/yihongyuelan/article/details/14000363

这个是光晕半径。也就是当我们按下并移动时，跟随我们手指移动的那一小团小白点。在4.0中是handleDrawable会跟着移动，4.2中但出发了onTouchEvent后，handleDrawable就会消失，取而代之的是出现一小圈小白点，也就是这里的光晕半径。

GlowPadWrapper 主要实现GlowPadView.OnTriggerListener 接口，负责GlowPadView发送**onGrabbed**、 **onReleased**、**onTrigger**事件的回调处理

GlowPadView

```
    @Override
    protected void onDraw(Canvas canvas) {
        mPointCloud.draw(canvas);
        mOuterRing.draw(canvas);
        final int ntargets = mTargetDrawables.size();
        for (int i = 0; i < ntargets; i++) {
            TargetDrawable target = mTargetDrawables.get(i);
            if (target != null) {
                target.draw(canvas);
            }
        }
        mHandleDrawable.draw(canvas);
    }
```

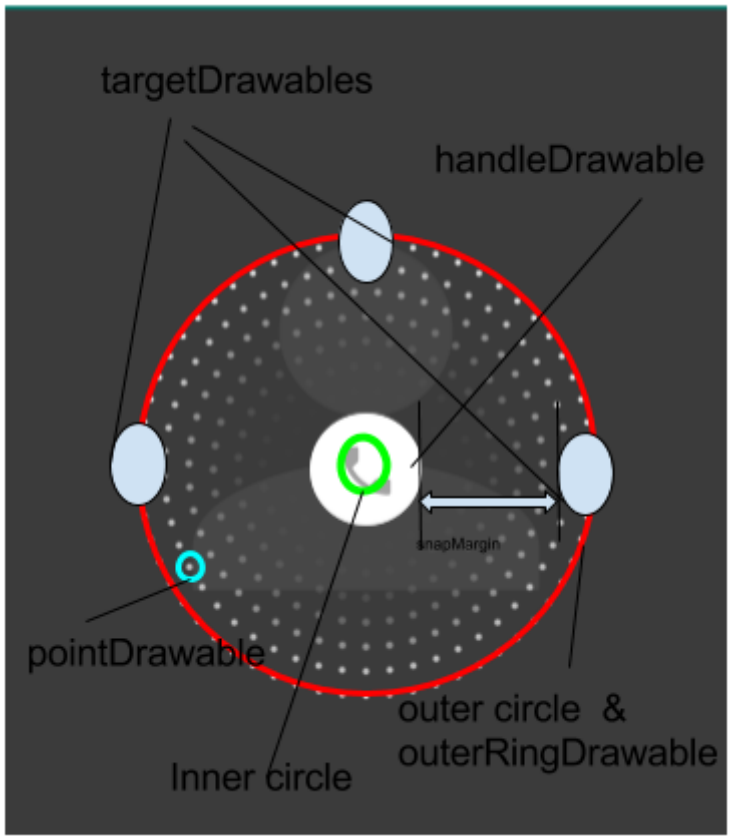
-绘制PointCloud区域中的点

-绘制外环圈

-绘制三个target图标

-绘制handle图标

绘制PointCloud区域中的点



1.先计算出所有点的向量

```
public void makePointCloud(float innerRadius, float outerRadius) {
    if (innerRadius == 0) {
        Log.w(TAG, "Must specify an inner radius");
        return;
    }
    mOuterRadius = outerRadius;
    mPointCloud.clear();
    final float pointAreaRadius = (outerRadius - innerRadius);
    final float ds = (2.0f * PI * innerRadius / INNER_POINTS);
    final int bands = (int) Math.round(pointAreaRadius / ds);
    final float dr = pointAreaRadius / bands;
    float r = innerRadius;
    for (int b = 0; b <= bands; b++, r += dr) {
        float circumference = 2.0f * PI * r;
        final int pointsInBand = (int) (circumference / ds);
        float eta = PI/2.0f;
        float dEta = 2.0f * PI / pointsInBand;
        for (int i = 0; i < pointsInBand; i++) {
            float x = r * (float) Math.cos(eta);
            float y = r * (float) Math.sin(eta);
            eta += dEta;
            mPointCloud.add(new Point(x, y, r));
        }
    }
}
```

2.画出所有点

```
public void draw(Canvas canvas) {
    ArrayList<Point> points = mPointCloud;
    canvas.save(Canvas.MATRIX_SAVE_FLAG);
    canvas.scale(mScale, mScale, mCenterX, mCenterY);
    for (int i = 0; i < points.size(); i++) {
        Point point = points.get(i);
        final float pointSize = interp(MAX_POINT_SIZE, MIN_POINT_SIZE,
                point.radius / mOuterRadius);
        final float px = point.x + mCenterX;
        final float py = point.y + mCenterY;
        int alpha = getAlphaForPoint(point);

        if (alpha == 0) continue;

        if (mDrawable != null) {
            canvas.save(Canvas.MATRIX_SAVE_FLAG);
            final float cx = mDrawable.getIntrinsicWidth() * 0.5f;
            final float cy = mDrawable.getIntrinsicHeight() * 0.5f;
            final float s = pointSize / MAX_POINT_SIZE;
            canvas.scale(s, s, px, py);
            canvas.translate(px - cx, py - cy);
            mDrawable.setAlpha(alpha);
            mDrawable.draw(canvas);
            canvas.restore();
        } else {
            mPaint.setAlpha(alpha);
            canvas.drawCircle(px, py, pointSize, mPaint);
        }
    }
    canvas.restore();
}
```

-绘制外环圈

在构造GlowPadView对象时，会构造外环圈对象，该对象的宽高是在outerRingDrawable属性中配置了，姑且叫做理想中的最小宽高。但由于在实际中上层布局会对下层布局的宽高进行测量，会有误差，所以实践中的宽高往往会和理想中的最小宽高有一个scale factor（比例系数）

```
mOuterRing = new TargetDrawable(res,
        getResourceId(a, R.styleable.GlowPadView_outerRingDrawable), 1);
```

measure

所以接下的measure阶段钟的getSuggestedMiniMumWidth（理想中的最小宽度）和getScaledSuggestedMiniMumWidth（通过比例系数计算出的宽度）就不然理解了。

computInsets 计算整个布局是水平右对齐、水平左对齐、水平居中，还是垂直右对齐、垂直 左对齐、垂直居中

```
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    final int minimumWidth = getSuggestedMinimumWidth();
    final int minimumHeight = getSuggestedMinimumHeight();
    int computedWidth = resolveMeasured(widthMeasureSpec, minimumWidth);
    int computedHeight = resolveMeasured(heightMeasureSpec, minimumHeight);

    mRingScaleFactor = computeScaleFactor(minimumWidth, minimumHeight,
            computedWidth, computedHeight);

    int scaledWidth = getScaledSuggestedMinimumWidth();
    int scaledHeight = getScaledSuggestedMinimumHeight();

    computeInsets(computedWidth - scaledWidth, computedHeight - scaledHeight);
    setMeasuredDimension(computedWidth, computedHeight);
}
```

layout

设置外环圈的中心点，不过如果是初次显示出来电界面的话，外环圈是不会显示出来的只有在手指滑动是才会显示，隐藏的外环圈的方法hideTargets

```java
@Override
protected void onLayout(boolean changed, int left, int top, int right, int bottom) {
    super.onLayout(changed, left, top, right, bottom);
    final int width = right - left;
    final int height = bottom - top;

    // Target placement width/height. This puts the targets on the greater of the ring
    // width or the specified outer radius.
    final float placementWidth = getRingWidth();
    final float placementHeight = getRingHeight();
    float newWaveCenterX = mHorizontalInset
            + (mMaxTargetWidth + placementWidth) / 2;
    float newWaveCenterY = mVerticalInset
            + (mMaxTargetHeight + placementHeight) / 2;

    if (mInitialLayout) {
        stopAndHideWaveAnimation();
        hideTargets(false, false);
        mInitialLayout = false;
    }

    mOuterRing.setPositionX(newWaveCenterX);
    mOuterRing.setPositionY(newWaveCenterY);

    mPointCloud.setScale(mRingScaleFactor);

    mHandleDrawable.setPositionX(newWaveCenterX);
    mHandleDrawable.setPositionY(newWaveCenterY);

    updateTargetPositions(newWaveCenterX, newWaveCenterY);
    updatePointCloudPosition(newWaveCenterX, newWaveCenterY);
    updateGlowPosition(newWaveCenterX, newWaveCenterY);

    mWaveCenterX = newWaveCenterX;
    mWaveCenterY = newWaveCenterY;

    if (DEBUG) dump();
}
```

hideTargets

通过动画隐藏

```
private void hideTargets(boolean animate, boolean expanded) {
    mTargetAnimations.cancel();
    // Note: these animations should complete at the same time so that we can swap out
    // the target assets asynchronously from the setTargetResources() call.
    mAnimatingTargets = animate;
    final int duration = animate ? HIDE_ANIMATION_DURATION : 0;
    final int delay = animate ? HIDE_ANIMATION_DELAY : 0;

    final float targetScale = expanded ?
            TARGET_SCALE_EXPANDED : TARGET_SCALE_COLLAPSED;
    final int length = mTargetDrawables.size();
    final TimeInterpolator interpolator = Ease.Cubic.easeOut;
    for (int i = 0; i < length; i++) {
        TargetDrawable target = mTargetDrawables.get(i);
        target.setState(TargetDrawable.STATE_INACTIVE);
        mTargetAnimations.add(Tweener.to(target, duration,
                "ease", interpolator,
                "alpha", 0.0f,
                "scaleX", targetScale,
                "scaleY", targetScale,
                "delay", delay,
                "onUpdate", mUpdateListener));
    }

    float ringScaleTarget = expanded ?
            RING_SCALE_EXPANDED : RING_SCALE_COLLAPSED;
    ringScaleTarget *= mRingScaleFactor;
    mTargetAnimations.add(Tweener.to(mOuterRing, duration,
            "ease", interpolator,
            "alpha", 0.0f,
            "scaleX", ringScaleTarget,
            "scaleY", ringScaleTarget,
            "delay", delay,
            "onUpdate", mUpdateListener,
            "onComplete", mTargetUpdateListener));

    mTargetAnimations.start();
}
```

draw

```
@Override
protected void onDraw(Canvas canvas) {
    mPointCloud.draw(canvas);
    mOuterRing.draw(canvas);
    final int ntargets = mTargetDrawables.size();
    for (int i = 0; i < ntargets; i++) {
        TargetDrawable target = mTargetDrawables.get(i);
        if (target != null) {
            target.draw(canvas);
        }
    }
    mHandleDrawable.draw(canvas);
}
```

```java
public void draw(Canvas canvas) {
    if (mDrawable == null || !mEnabled) {
        return;
    }
    canvas.save(Canvas.MATRIX_SAVE_FLAG);
    canvas.scale(mScaleX, mScaleY, mPositionX, mPositionY);
    canvas.translate(mTranslationX + mPositionX, mTranslationY + mPositionY);
    canvas.translate(-0.5f * getWidth(), -0.5f * getHeight());
    mDrawable.setAlpha((int) Math.round(mAlpha * 255f));
    mDrawable.draw(canvas);
    canvas.restore();
}
```

-绘制三个target图标

getVaule查询targetDrawables对应的属性值，并赋值给outValue对象。该对象的resourceId是资源的id，其中该资源是大小为4的数组

```java
TypedValue outValue = new TypedValue();

// Read array of target drawables
if (a.getValue(R.styleable.GlowPadView_targetDrawables, outValue)) {
    internalSetTargetResources(outValue.resourceId);
}
if (mTargetDrawables == null || mTargetDrawables.size() == 0) {
    throw new IllegalStateException("Must specify at least one target drawable");
}
```

```java
private void internalSetTargetResources(int resourceId) {
    final ArrayList<TargetDrawable> targets = loadDrawableArray(resourceId);
    mTargetDrawables = targets;
    mTargetResourceId = resourceId;

    int maxWidth = mHandleDrawable.getWidth();
    int maxHeight = mHandleDrawable.getHeight();
    final int count = targets.size();
    for (int i = 0; i < count; i++) {
        TargetDrawable target = targets.get(i);
        maxWidth = Math.max(maxWidth, target.getWidth());
        maxHeight = Math.max(maxHeight, target.getHeight());
    }
    if (mMaxTargetWidth != maxWidth || mMaxTargetHeight != maxHeight) {
        mMaxTargetWidth = maxWidth;
        mMaxTargetHeight = maxHeight;
        requestLayout(); // required to resize layout and call updateTargetPositions()
    } else {
        updateTargetPositions(mWaveCenterX, mWaveCenterY);
        updatePointCloudPosition(mWaveCenterX, mWaveCenterY);
    }
}
```

loadDrawableArry

```
private ArrayList<TargetDrawable> loadDrawableArray(int resourceId) {
    Resources res = getContext().getResources();
    TypedArray array = res.obtainTypedArray(resourceId);
    final int count = array.length();
    ArrayList<TargetDrawable> drawables = new ArrayList<TargetDrawable>(count);
    for (int i = 0; i < count; i++) {
        TypedValue value = array.peekValue(i);
        TargetDrawable target = new TargetDrawable(res, value != null ? value.resourceId : 0, 3);
        drawables.add(target);
    }
    array.recycle();
    return drawables;
}
```

updatePointCloudPostion

```
private void updateTargetPositions(float centerX, float centerY) {
    // Reposition the target drawables if the view changed.
    ArrayList<TargetDrawable> targets = mTargetDrawables;
    final int size = targets.size();
    final float alpha = (float) (-2.0f * Math.PI / size);
    for (int i = 0; i < size; i++) {
        final TargetDrawable targetIcon = targets.get(i);
        final float angle = alpha * i;
        targetIcon.setPositionX(centerX);
        targetIcon.setPositionY(centerY);
        targetIcon.setX(getRingWidth() / 2 * (float) Math.cos(angle));
        targetIcon.setY(getRingHeight() / 2 * (float) Math.sin(angle));
    }
}
```

measure

layout

hideTargets

```
for (int i = 0; i < length; i++) {
    TargetDrawable target = mTargetDrawables.get(i);
    target.setState(TargetDrawable.STATE_INACTIVE);
    mTargetAnimations.add(Tweener.to(target, duration,
            "ease", interpolator,
            "alpha", 0.0f,
            "scaleX", targetScale,
            "scaleY", targetScale,
            "delay", delay,
            "onUpdate", mUpdateListener));
}
```

```
private void updateTargetPositions(float centerX, float centerY) {
    // Reposition the target drawables if the view changed.
    ArrayList<TargetDrawable> targets = mTargetDrawables;
    final int size = targets.size();
    final float alpha = (float) (-2.0f * Math.PI / size);
    for (int i = 0; i < size; i++) {
        final TargetDrawable targetIcon = targets.get(i);
        final float angle = alpha * i;
        targetIcon.setPositionX(centerX);
        targetIcon.setPositionY(centerY);
        targetIcon.setX(getRingWidth() / 2 * (float) Math.cos(angle));
        targetIcon.setY(getRingHeight() / 2 * (float) Math.sin(angle));
    }
}
```

draw

```
final int ntargets = mTargetDrawables.size();
for (int i = 0; i < ntargets; i++) {
    TargetDrawable target = mTargetDrawables.get(i);
    if (target != null) {
        target.draw(canvas);
    }
}
```

同绘制外环圈一样。

-绘制handle图标

通过peekVaule 得到的handle对象，该对象的resourceId是资源的id

```
TypedValue handle = a.peekValue(R.styleable.GlowPadView_handleDrawable);
setHandleDrawable(handle != null ? handle.resourceId : R.drawable.ic_incall_audio_handle);
```

```
public void setHandleDrawable(int resourceId) {
    /// M: Modify this for Incoming call ui display. @{
    // Set handle drawable only if it has changed.
    if (mHandleDrawable == null || mHandleDrawable.getResourceId() != resourceId) {
        mHandleDrawable = new TargetDrawable(getResources(), resourceId, 2);
        mHandleDrawable.setState(TargetDrawable.STATE_INACTIVE);
    }
    /// @}
}
```

measure

layout

```
mHandleDrawable.setPositionX(newWaveCenterX);
mHandleDrawable.setPositionY(newWaveCenterY);
```

draw

```
mHandleDrawable.draw(canvas);
```

动画

AnimationBundle类是Tweener的集合，里面是有一系列的动画效果，而Tweener是属性动画ObjectAnimator的封装器，一个Tweener对象持有一个属性动画对象，而一个Tweener对象和TargetDrawable对象是键值对的关系，也是一对一。所以在界面上我们就可以看到，每个TargetDrawable都有属于自己的动画

```java
private void hideTargets(boolean animate, boolean expanded) {
    mTargetAnimations.cancel();
    // Note: these animations should complete at the same time so that we can swap out
    // the target assets asynchronously from the setTargetResources() call.
    mAnimatingTargets = animate;
    final int duration = animate ? HIDE_ANIMATION_DURATION : 0;
    final int delay = animate ? HIDE_ANIMATION_DELAY : 0;

    final float targetScale = expanded ?
            TARGET_SCALE_EXPANDED : TARGET_SCALE_COLLAPSED;
    final int length = mTargetDrawables.size();
    final TimeInterpolator interpolator = Ease.Cubic.easeOut;
    for (int i = 0; i < length; i++) {
        TargetDrawable target = mTargetDrawables.get(i);
        target.setState(TargetDrawable.STATE_INACTIVE);
        mTargetAnimations.add(Tweener.to(target, duration,
                "ease", interpolator,
                "alpha", 0.0f,
                "scaleX", targetScale,
                "scaleY", targetScale,
                "delay", delay,
                "onUpdate", mUpdateListener));
    }

    float ringScaleTarget = expanded ?
            RING_SCALE_EXPANDED : RING_SCALE_COLLAPSED;
    ringScaleTarget *= mRingScaleFactor;
    mTargetAnimations.add(Tweener.to(mOuterRing, duration,
            "ease", interpolator,
            "alpha", 0.0f,
            "scaleX", ringScaleTarget,
            "scaleY", ringScaleTarget,
            "delay", delay,
            "onUpdate", mUpdateListener,
            "onComplete", mTargetUpdateListener));

    mTargetAnimations.start();
}
```

http://robertpenner.com/easing/penner_chapter7_tweening.pdf

http://robertpenner.com/easing/

Ease类定义了Interpolator的种类

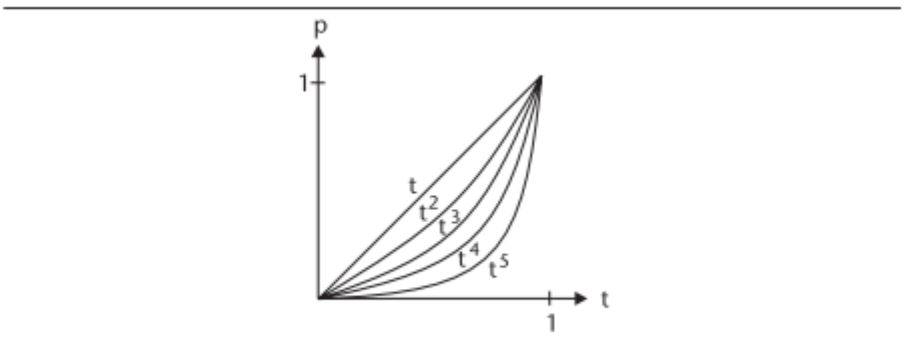


FIGURE 7-11

Graphs of t, t^2, t^3, t^4, and t^5 easing

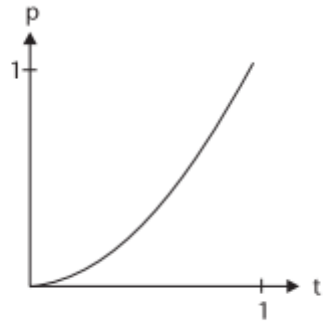


FIGURE 7-12

Graph of sinusoidal easing

交互

只要当使用者点击屏幕就会触动GlowPadView的onTouchEvent方法的回调，最终的结果就是触发定义在GlowPadView类里面的接口OnTriggerListener的方法。

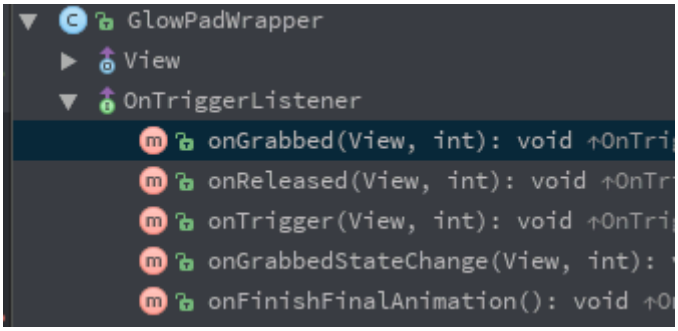
```java
@Override
public boolean onTouchEvent(MotionEvent event) {
    final int action = event.getActionMasked();
    boolean handled = false;
    switch (action) {
        case MotionEvent.ACTION_POINTER_DOWN:
        case MotionEvent.ACTION_DOWN:
            if (DEBUG) Log.v(TAG, "*** DOWN ***");
            handleDown(event);
            handleMove(event);
            handled = true;
            break;

        case MotionEvent.ACTION_MOVE:
            if (DEBUG) Log.v(TAG, "*** MOVE ***");
            handleMove(event);
            handled = true;
            break;

        case MotionEvent.ACTION_POINTER_UP:
        case MotionEvent.ACTION_UP:
            if (DEBUG) Log.v(TAG, "*** UP ***");
            handleMove(event);
            handleUp(event);
            handled = true;
            break;

        case MotionEvent.ACTION_CANCEL:
            if (DEBUG) Log.v(TAG, "*** CANCEL ***");
            handleMove(event);
            handleCancel(event);
            handled = true;
            break;
    }
    invalidate();
    return handled ? true : super.onTouchEvent(event);
}
```

为了简明GlowPadView的任务，就让子类GlowPadWrapper去实现OnTriggerListener接口。这样GlowPadWrapper就可以专注于触发事件的处理了。



GlowPadWrapper有五个回调方法却只实现了三个方法的功能、

```java
@Override
public void onGrabbed(View v, int handle) {
    Log.d(this, "onGrabbed()");
    stopPing();
}

@Override
public void onReleased(View v, int handle) {
    Log.d(this, "onReleased()");
    if (mTargetTriggered) {
        mTargetTriggered = false;
    } else {
        startPing();
    }
}

@Override
public void onTrigger(View v, int target) {
    Log.d(this, "onTrigger() view=" + v + " target=" + target);
    final int resId = getResourceIdForTarget(target);
    if (resId == R.drawable.ic_lockscreen_answer) {
        mAnswerFragment.onAnswer(VideoProfile.STATE_AUDIO_ONLY, getContext());
        mTargetTriggered = true;
    } else if (resId == R.drawable.ic_lockscreen_decline) {
        mAnswerFragment.onDecline(getContext());
        mTargetTriggered = true;
    } else if (resId == R.drawable.ic_lockscreen_text) {
        mAnswerFragment.onText();
        mTargetTriggered = true;
    } else if (resId == R.drawable.ic_videocam || resId == R.drawable.ic_lockscreen_answer_video) {
        mAnswerFragment.onAnswer(mVideoState, getContext());
        mTargetTriggered = true;
    } else if (resId == R.drawable.ic_lockscreen_decline_video) {
        mAnswerFragment.onDeclineUpgradeRequest(getContext());
        mTargetTriggered = true;
    } else {
        // Code should never reach here.
        Log.e(this, "Trigger detected on unhandled resource. Skipping.");
    }
}
```