

灵动的锦鲤鱼

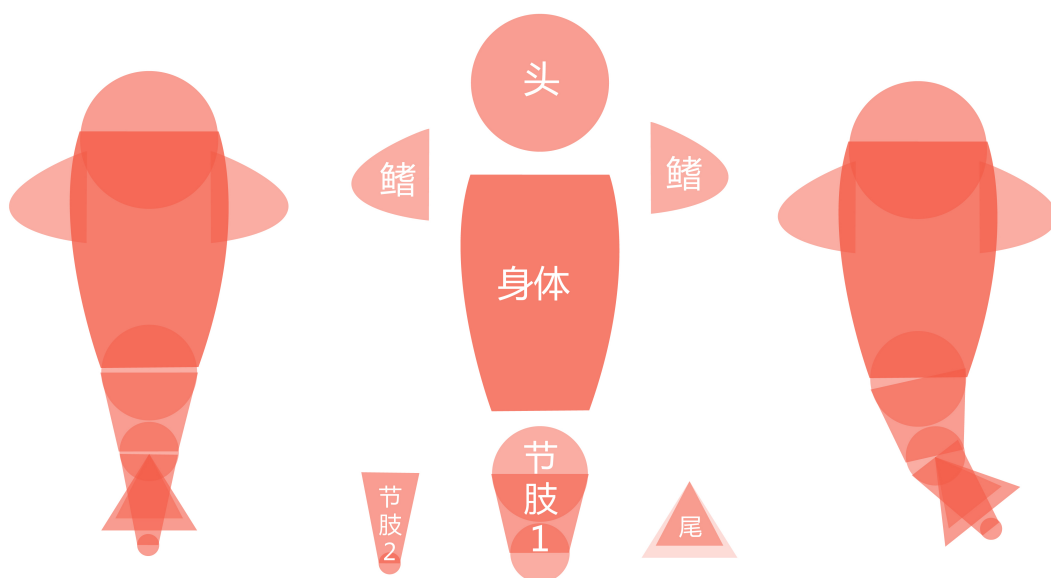
1.绘制小鱼

1-1.技术点

绘制实现主要用到技术：

1. 自定义Drawable动画
2. Android的坐标及角度
3. 正余弦函数的使用以及角度和弧度的转换

1-2.分解图



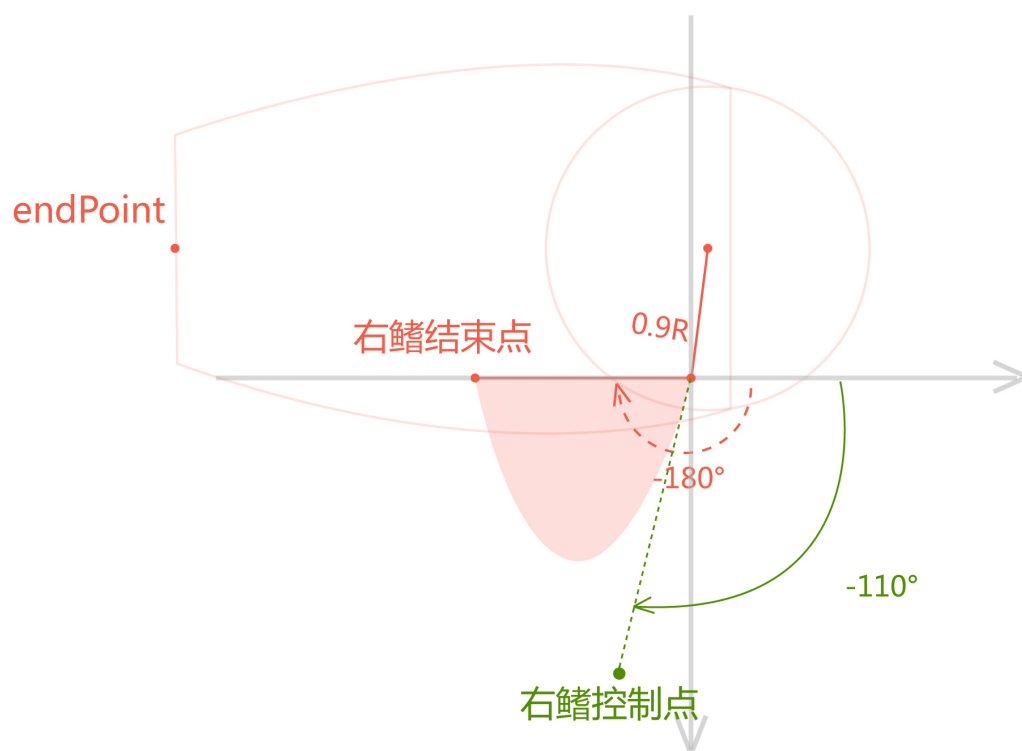
分析效果：

1. 小鱼的身体各个部件都是简单的半透明几何图形
2. 各个部件都可以活动
3. 从头到尾方向的部件摆动幅度越来越大、频率越来越高

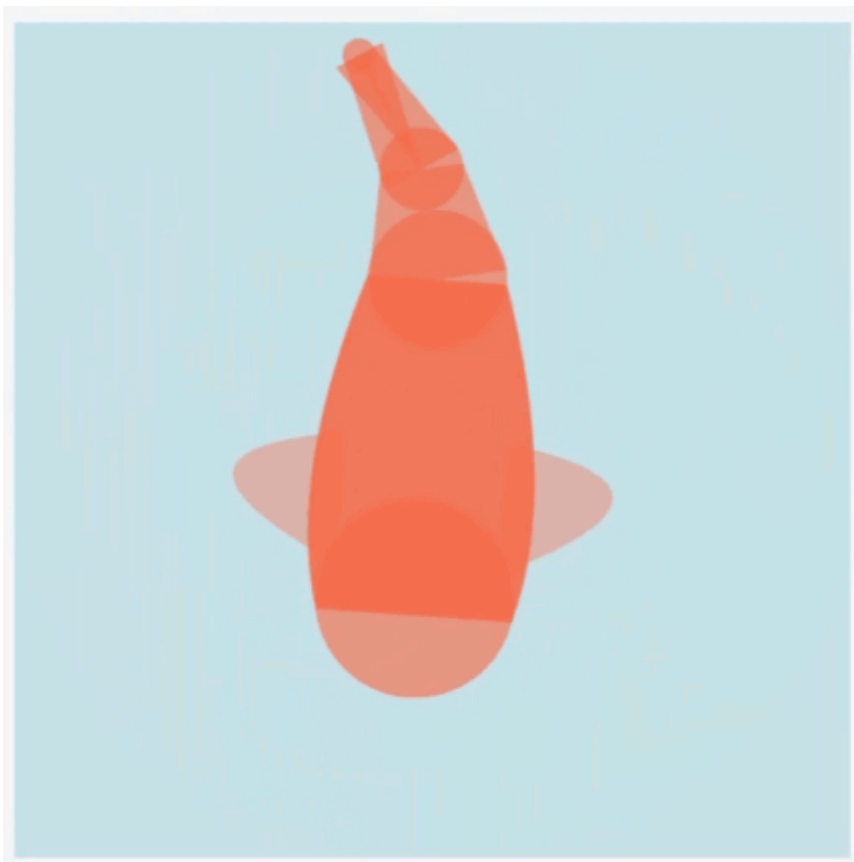
1-3.鱼鳍的绘制

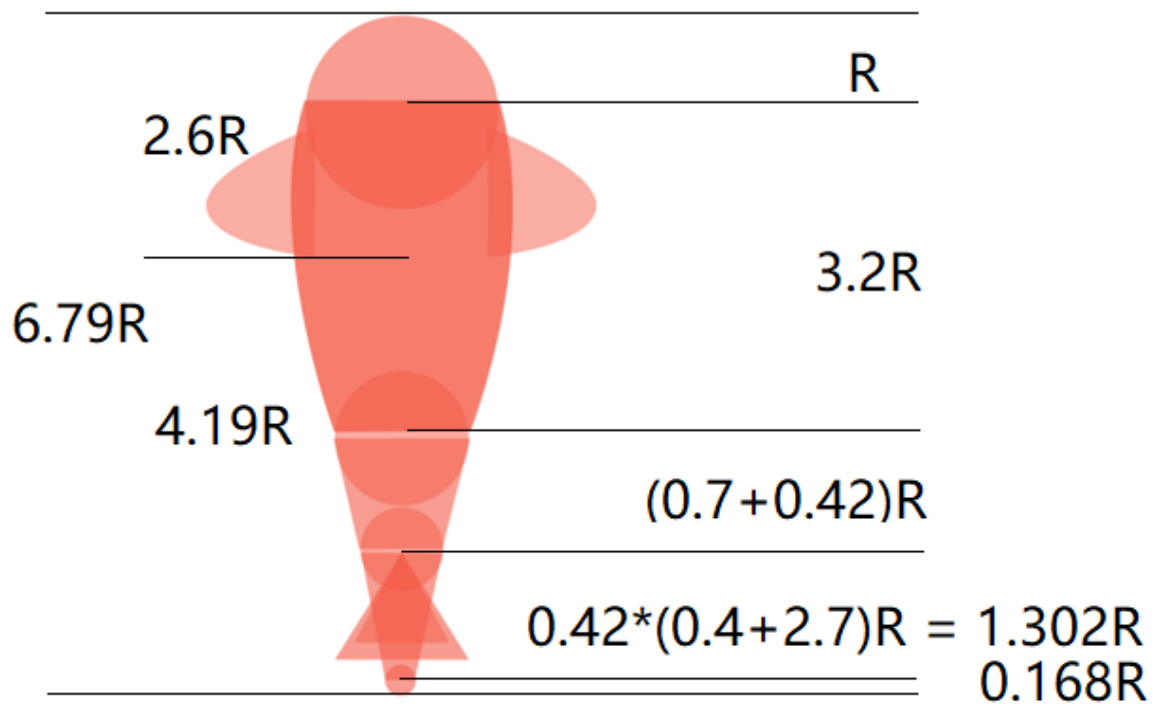
鱼的绘制必须要有个相对点，这样移动这个点，就能移动整个鱼了。下面只讲解鱼鳍的绘制，其他类似。

先假设鱼身角度为 0° ，即头朝向X轴正方向。通过重心点以及第一节身长的一半的长度，以及角度即可计算出头部的圆心坐标，然后再以头部圆心坐标和 $0.9R$ 的长度，顺时针旋转 80° 确定右边鱼鳍的坐标点。



1-4. 鱼旋转需要的空间





宽高值: $4.19R \times 2 = 8.38R$

1-5.源码

1-5-1.自定义Drawable, 重写方法

```
public class FishDrawable extends Drawable {

    /**
     * 绘制, 类似自定义View中的onDraw方法
     */
    @Override
    public void draw(@NonNull Canvas canvas) {
        makeFish(canvas);
    }

    /**
     * 设置透明度的方法
     */
    @Override
    public void setAlpha(int alpha) {
        // 设置Drawable的透明度, 一般情况下将此alpha值设置给Paint
        mPaint.setAlpha(alpha);
    }

    /**
     * 设置了一个颜色过滤器, 那么在绘制出来之前, 被绘制内容的每一个像素都会被颜色过滤器改变
     */
    @Override
    public void setColorFilter(@Nullable ColorFilter colorFilter) {
        // 设置颜色滤镜, 一般情况下将此值设置给Paint
        mPaint.setColorFilter(colorFilter);
    }
}
```

```

/**
 * 这个值，可以根据setAlpha中设置的值进行调整。比如，alpha == 0时设置为
PixelFormat.TRANSPARENT。
 * 在alpha == 255时设置为PixelFormat.OPAQUE。在其他时候设置为
PixelFormat.TRANSLUCENT。
 * PixelFormat.OPAQUE：便是完全不透明，遮盖在他下面的所有内容
 * PixelFormat.TRANSPARENT：透明，完全不显示任何东西
 * PixelFormat.TRANSLUCENT：只有绘制的地方才覆盖底下的内容
 */
@Override
public int getOpacity() {
    return PixelFormat.TRANSLUCENT;
}

/**
 * 在View使用wrap_content的时候，设置固定宽度，默认为-1
 */
@Override
public int getIntrinsicWidth() {
    return (int) (8.38f * HEAD_RADIUS);
}

/**
 * 在View使用wrap_content的时候，设置固定高度，默认为-1
 */
@Override
public int getIntrinsicHeight() {
    return (int) (8.38f * HEAD_RADIUS);
}
}

```

1-5-2.初始化

```

private Path mPath;
private Paint mPaint;

//转弯更自然的重心(身体的中心点)
private PointF middlePoint;

public FishDrawable() {
    init();
}

private void init() {
    // 路径
    mPath = new Path();
    // 画笔
    mPaint = new Paint();
    // 抗锯齿
    mPaint.setAntiAlias(true);
    // 画笔类型填充
    mPaint.setStyle(Paint.Style.FILL);
    // 防抖
    mPaint.setDither(true);
    // 设置颜色
    mPaint.setColor(Color.argb(OTHER_ALPHA, 244, 92, 71));
    // 与Point一样，只是坐标为浮点数
}

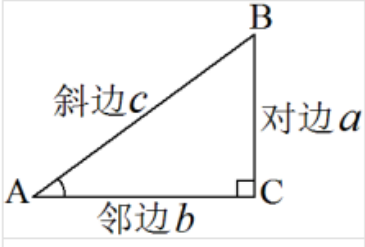
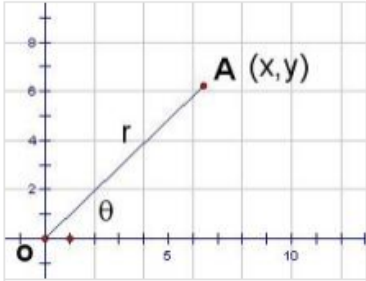
```

```
// 重心位于整个控件的中心，保证鱼旋转的空间
```

```
middlePoint = new PointF(4.18f * HEAD_RADIUS, 4.18f * HEAD_RADIUS);
```

```
}
```

1-5-3.计算坐标

	锐角三角函数	任意角三角函数
图形	 <p>直角三角形</p>	 <p>任意角三角函数</p>
正弦 (sin)	$\sin A = \frac{a}{c}$	$\sin \theta = \frac{y}{r}$
余弦 (cos)	$\cos A = \frac{b}{c}$	$\cos \theta = \frac{x}{r}$
正切 (tan或tg)	$\tan A = \frac{a}{b}$	$\tan \theta = \frac{y}{x}$
余切 (cot或ctg)	$\cot A = \frac{b}{a}$	$\cot \theta = \frac{x}{y}$
正割 (sec)	$\sec A = \frac{c}{b}$	$\sec \theta = \frac{r}{x}$
余割 (csc)	$\csc A = \frac{c}{a}$	$\csc \theta = \frac{r}{y}$

- $\sin A = a/c \rightarrow \sin A * c = a \rightarrow$ 得到B点的y坐标
- $\cos A = b/c \rightarrow \cos A * c = b \rightarrow$ 得到B点的x坐标
- Math.sin()、Math.cos()的参数是弧度。坐标是按数学中的坐标。
- Math.toRadians() 将角度转成弧度。
- 圆是360度，也是 2π 弧度，即 $360^\circ = 2\pi$

注意点：

- 与数学坐标不同的是Android的坐标中Y轴正方向是朝下的，但是角度却和平面直角坐标系的计算方法一样，即原点指向X轴正方向为 0° ，正角度是逆时针旋转，负角度是顺时针旋转。
- 那么问题就来了：坐标系不同，角度转动方式却一样，这就导致Math.sin(α)与安卓中的sin值符号相反。
- 所以我们需要将Math.sin(α)取反，根据下面的两个公式，我们只需要将 α 加上或者减去 180° 即可。

```
/**
 * 输入起点、长度、旋转角度计算终点
 *
 * @param startPoint 起点
 * @param length 长度
 * @param angle 旋转角度
 * @return 计算结果点
 */
public static PointF calculatPoint(PointF startPoint, float length, float angle)
{
    // Math.toRadians 角度转弧度 --- sin\cos的参数是弧度制
    float deltax = (float) Math.cos(Math.toRadians(angle)) * length;
    // 符合Android坐标的y轴朝下的标准
```

```

float deltaY = (float) Math.sin(Math.toRadians(angle - 180)) * length;
return new PointF(startPoint.x + deltaX, startPoint.y + deltaY);
}

```

1-5-4.画鱼

```

// 透明度
private static final int OTHER_ALPHA = 110;
// 鱼身的透明度
private static final int BODY_ALPHA = 160;

// 鱼的初始角度,0表示朝x轴正方向,90表示朝y轴负方向
private final float fishStartAngle = 90;

//转弯更自然的重心(身体的中心点)
private PointF middlePoint;

//鱼头圆的圆心
private PointF headPoint;

/**
 * 鱼的各部位长度
 */
// 鱼头半径
private static final float HEAD_RADIUS = 150;
// 身体的长度
private static final float BODY_LENGTHT = HEAD_RADIUS * 3.2f;
// -----鱼鳍-----
// 寻找鱼鳍开始点的线长
private static final float FIND_FINS_LENGTH = HEAD_RADIUS * 0.9f;
// 鱼鳍的长度
private final float FINS_LENGTH = HEAD_RADIUS * 1.3f;
// -----鱼尾-----
// 尾部大圆的半径(圆心就是身体底部的中点)
private final float BIG_CIRCLE_RADIUS = HEAD_RADIUS * 0.7f;
// --寻找尾部中圆圆心的线长
private final float FIND_MIDDLE_CIRCLE_LENGTH = BIG_CIRCLE_RADIUS * (0.6f + 1);
// 尾部中圆的半径
private final float MIDDLE_CIRCLE_RADIUS = BIG_CIRCLE_RADIUS * 0.6f;
// --寻找尾部小圆圆心的线长
private final float FIND_SMALL_CIRCLE_LENGTH = MIDDLE_CIRCLE_RADIUS * (0.4f + 2.7f);
// 尾部小圆的半径
private final float SMALL_CIRCLE_RADIUS = MIDDLE_CIRCLE_RADIUS * 0.4f;
// --寻找大三角形底边中心点的线长
private final float FIND_TRIANGLE_LENGTH = FIND_SMALL_CIRCLE_LENGTH;

/**
 * 画鱼
 */
private void makeFish(Canvas canvas) {
    float fishAngle = fishStartAngle;

    // 画鱼头的圆心坐标
    headPoint = calculatPoint(middlePoint, BODY_LENGTHT / 2, fishAngle);
    // 画鱼头

```

```

        canvas.drawCircle(headPoint.x, headPoint.y, HEAD_RADIUS, mPaint);

        // 画右鳍的起点坐标
        PointF rightFinsPoint = calculatPoint(headPoint, FIND_FINS_LENGTH, fishAngle
- 110);
        // 画右鳍
        makeFins(canvas, rightFinsPoint, fishAngle, true);

        // 画左鳍的起点坐标
        PointF leftFinsPoint = calculatPoint(headPoint, FIND_FINS_LENGTH, fishAngle
+ 110);
        // 画左鳍
        makeFins(canvas, leftFinsPoint, fishAngle, false);

        // 身体底部的中心点
        PointF bodyBottomCenterPoint = calculatPoint(headPoint, BODY_LENGTH,
fishAngle - 180);
        // 节肢1
        PointF tailMainPoint = makeSegment(canvas, bodyBottomCenterPoint,
BIG_CIRCLE_RADIUS,
            FIND_MIDDLE_CIRCLE_LENGTH, MIDDLE_CIRCLE_RADIUS, fishAngle, true);
        // 节肢2
        makeSegment(canvas, tailMainPoint, MIDDLE_CIRCLE_RADIUS,
FIND_SMALL_CIRCLE_LENGTH,
            SMALL_CIRCLE_RADIUS, fishAngle, false);

        // 尾部三角形
        float triangleHalfLength = MIDDLE_CIRCLE_RADIUS + HEAD_RADIUS / 5 * 3;
        makeTriangle(canvas, tailMainPoint, FIND_TRIANGLE_LENGTH,
triangleHalfLength, fishAngle);
        makeTriangle(canvas, tailMainPoint, FIND_TRIANGLE_LENGTH - 10,
            triangleHalfLength - 20, fishAngle);

        // 鱼身
        makeBody(canvas, bodyBottomCenterPoint, fishAngle);
    }
}

```

1-5-5.画鱼鳍

```

/**
 * 画鱼鳍
 *
 * @param canvas
 * @param startPoint
 * @param isRightFins 绘制的是否是右鱼鳍, true是的
 */
private void makeFins(Canvas canvas, PointF startPoint, float fishAngle, boolean
isRightFins) {
    // 鱼鳍二阶贝塞尔曲线的控制点
    float controlAngle = 115; // 鱼鳍三角控制角度
    // 鱼鳍结束点
    PointF endPoint = calculatPoint(startPoint, FINS_LENGTH, fishAngle - 180);
    // 控制点
    PointF controlPoint = calculatPoint(startPoint, FINS_LENGTH * 1.8f,
        isRightFins ? fishAngle - controlAngle : fishAngle + controlAngle);

    mPath.reset();
}

```

```

// 移动到画鱼鳍的起点坐标
mPath.moveTo(startPoint.x, startPoint.y);
// 二阶贝塞尔曲线
mPath.quadTo(controlPoint.x, controlPoint.y, endPoint.x, endPoint.y);
canvas.drawPath(mPath, mPaint);
}

```

1-5-6.画节肢

```

/**
 * 画节肢1,2(这两个摆动的时候是分开的, 必须分开画)
 *
 * @param bottomCenterPoint 梯形上底的中心点
 * @param bigCircleRadius 梯形大圆的半径
 * @param findSmallCircleLength 寻找梯形小圆的线长
 * @param smallCircleRadius 梯形小圆的半径
 * @param hasBigCircle 节肢是否需要绘制大圆, true绘制
 * @return 计算节肢1的时候需要返回梯形小圆的圆心点, 这个是绘制节肢2和三角形的起始点
 */
private PointF makeSegment(Canvas canvas, PointF bottomCenterPoint, float
bigCircleRadius,
                                float findSmallCircleLength, float smallCircleRadius,
float fishAngle,
                                boolean hasBigCircle) {
    float segmentAngle = fishAngle;
    // 梯形上底的中心点
    PointF upperCenterPoint = calculatPoint(bottomCenterPoint,
findSmallCircleLength, segmentAngle - 180);

    // 梯形的四个点
    PointF bottomLeftPoint = calculatPoint(bottomCenterPoint, bigCircleRadius,
segmentAngle + 90);
    PointF bottomRightPoint = calculatPoint(bottomCenterPoint, bigCircleRadius,
segmentAngle - 90);
    PointF upperLeftPoint = calculatPoint(upperCenterPoint, smallCircleRadius,
segmentAngle + 90);
    PointF upperRightPoint = calculatPoint(upperCenterPoint, smallCircleRadius,
segmentAngle - 90);

    if (hasBigCircle) {
        // 画大圈
        canvas.drawCircle(bottomCenterPoint.x, bottomCenterPoint.y,
bigCircleRadius, mPaint);
    }
    // 画小圈
    canvas.drawCircle(upperCenterPoint.x, upperCenterPoint.y, smallCircleRadius,
mPaint);
    // 画梯形
    mPath.reset();
    mPath.moveTo(bottomLeftPoint.x, bottomLeftPoint.y);
    mPath.lineTo(upperLeftPoint.x, upperLeftPoint.y);
    mPath.lineTo(upperRightPoint.x, upperRightPoint.y);
    mPath.lineTo(bottomRightPoint.x, bottomRightPoint.y);
    canvas.drawPath(mPath, mPaint);

    return upperCenterPoint;
}

```


1-5-7.绘制三角形

```
/**
 * 绘制三角形
 *
 * @param findTriangleLength 寻找三角形底边中心点的线长
 * @param triangleHalfLength 底边一半的长度
 */
private void makeTriangle(Canvas canvas, PointF startPoint, float
findTriangleLength,
                        float triangleHalfLength, float fishAngle) {
    // 与节肢2的摆幅一样
    float segmentAngle = fishAngle;

    // 三角形底边的中心点
    PointF centerPoint = calculatPoint(startPoint, findTriangleLength,
segmentAngle - 180);

    // 三角形底边的两点
    PointF leftPoint = calculatPoint(centerPoint, triangleHalfLength,
segmentAngle + 90);
    PointF rightPoint = calculatPoint(centerPoint, triangleHalfLength,
segmentAngle - 90);

    // 绘制三角形
    mPath.reset();
    mPath.moveTo(startPoint.x, startPoint.y);
    mPath.lineTo(leftPoint.x, leftPoint.y);
    mPath.lineTo(rightPoint.x, rightPoint.y);
    canvas.drawPath(mPath, mPaint);
}
```

1-5-8.画鱼身体

```
/**
 * 画鱼身体
 */
private void makeBody(Canvas canvas, PointF bodyBottomCenterPoint, float
fishAngle) {
    // 身体的四个点
    PointF topLeftPoint = calculatPoint(headPoint, HEAD_RADIUS, fishAngle + 80);
    PointF topRightPoint = calculatPoint(headPoint, HEAD_RADIUS, fishAngle -
80);
    PointF bottomLeftPoint = calculatPoint(bodyBottomCenterPoint,
BIG_CIRCLE_RADIUS, fishAngle + 90);
    PointF bottomRightPoint = calculatPoint(bodyBottomCenterPoint,
BIG_CIRCLE_RADIUS, fishAngle - 90);

    // 二阶贝塞尔曲线的控制点，决定鱼的胖瘦
    PointF contralLeft = calculatPoint(headPoint, BODY LENGHT * 0.56f, fishAngle
+ 130);
    PointF contralRight = calculatPoint(headPoint, BODY LENGHT * 0.56f,
fishAngle - 130);

    mPath.reset();
```

```

mPath.moveTo(topLeftPoint.x, topLeftPoint.y);
mPath.quadTo(contralLeft.x, contralLeft.y, bottomLeftPoint.x,
bottomLeftPoint.y);
mPath.lineTo(bottomRightPoint.x, bottomRightPoint.y);
mPath.quadTo(contralRight.x, contralRight.y, topRightPoint.x,
topRightPoint.y);
mPaint.setColor(Color.argb(BODY_ALPHA, 244, 92, 71));
canvas.drawPath(mPath, mPaint);
}

```

2.绘制动画



每一个部件摆动的角度不同。通过属性动画，给它一个变化的角度即可。

```

private final float CHANGE_VALUE = 2160f;
// 动画持续时间
private final int ANIMATOR_DURATION = 8 * 1000;

private void init() {
    ...

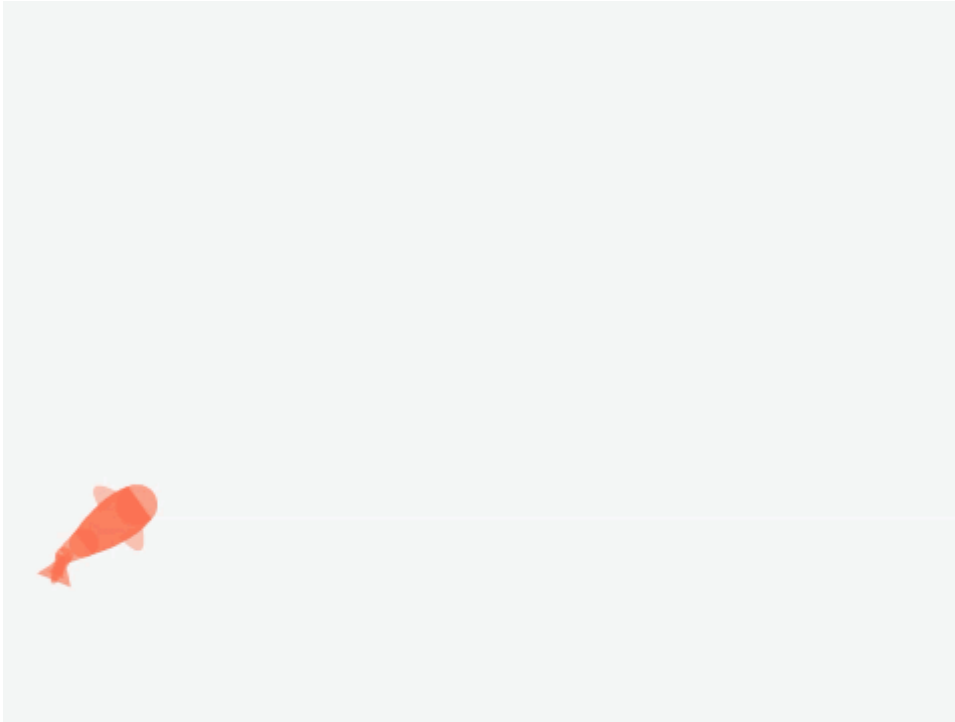
    ValueAnimator valueAnimator = ValueAnimator.ofFloat(0, CHANGE_VALUE); // 变化
    值
    valueAnimator.setDuration(ANIMATOR_DURATION); // 设置动画周期
    valueAnimator.setRepeatCount(ValueAnimator.INFINITE); // 设置循环次数，无限次
    valueAnimator.setRepeatMode(ValueAnimator.RESTART); // 设置循环模式
    valueAnimator.setInterpolator(new LinearInterpolator()); // 默认插值器为
    AccelerateDecelerateInterpolator
    valueAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
        @Override
        public void onAnimationUpdate(ValueAnimator animator) {
            currentValue = (float) animator.getAnimatedValue();
            invalidateSelf();
        }
    });
    valueAnimator.start();
}

```

因为鱼的摆动是周期性的，刚好可以用sin, cos去处理。例如鱼头的角度：

```
// 鱼头的角度
float fishAngle = (float) (fishStartAngle + Math.sin(Math.toRadians(currentValue
* 1.2)) * 4);
```

3.游动部分



3-1.水波纹的实现

直接通过属性动画实现。

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    x = event.getX();
    y = event.getY();

    // 水波纹的引擎
    ObjectAnimator rippleAnimator = ObjectAnimator.ofFloat(this, "radius",
        0f, 1f).setDuration(1000);
    rippleAnimator.start();

    return super.onTouchEvent(event);
}

@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);

    // 绘制水波纹
    mPaint.setAlpha(alpha);
    canvas.drawCircle(x, y, radius * 150, mPaint);

    invalidate();
}

public void setRadius(float radius) {
```

```
// 水波纹的透明度：从100到0
alpha = (int) (100 * (1 - radius));
// 水波纹的半径：从0到1变化
this.radius = radius;
}
```

3-2.鱼身的位移

A : 鱼头圆心 (控制点1)

O : 鱼身重心

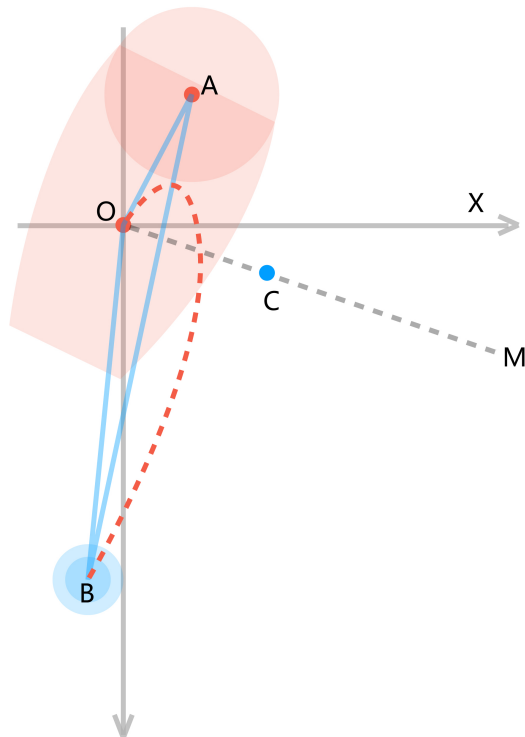
B : 手指点击处

C : 控制点2

OX : X轴方向

OM : $\angle AOB$ 中分线

OC长度=OA长度



1. 利用头部圆心、鱼身的重心以及点击点坐标来唯一确定一个特征三角形。
 2. 确定鱼身需要向左还是向右转弯，知道三角形内角AOB的大小，就知道转动的方向了。
- 向量夹角的定义：两相交直线所成的锐角或直角为两直线夹角。向量都有方向，两个向量正向的夹角就是平面向量的夹角，如 $\angle aob=60^\circ$ ，就是指向量oa与ob夹角为 60° ，而说向量ao与向量ob夹角，那就是 120° 了。向量夹角的范围是 $[0^\circ, 180^\circ]$ 。
 - 而向量夹角的余弦值等于= 向量的乘积/向量模的积。
 - 即向量的夹角公式： $\cos\theta = \frac{\text{向量a} \cdot \text{向量b}}{|\text{向量a}| \times |\text{向量b}|}$ 。

向量的夹角公式计算夹角 $\cos AOB = \frac{OA \cdot OB}{(|OA| \times |OB|)}$ 其中 $OA \cdot OB$ 是向量的数量积, 计算过程如下
 $OA = (Ax - Ox, Ay - Oy)$
 $OB = (Bx - Ox, By - Oy)$
 $OA \cdot OB = (Ax - Ox)(Bx - Ox) + (Ay - Oy)(By - Oy)$
 $|OA|$ 表示线段OA的模即OA的长度

通过向量夹角公式即可计算出控制点C坐标。

```
// 计算控制点
final float angle = includedAngle(fishMiddle, fishHead, touch);
float delta = calculatAngle(fishMiddle, fishHead);
PointF controlF = fishDrawable.calculatPoint(fishMiddle,
    1.6f * fishDrawable.HEAD_RADIUS, angle / 2 + delta);
```

```

/**
 * 开始点与结束点连成的线和x轴夹角
 */
public static float calculatAngle(PointF start, PointF end) {
    return includedAngle(start, new PointF(start.x + 1, start.y), end);
}

/**
 * 利用向量的夹角公式计算夹角
 *  $\cos AOB = (OA \cdot OB) / (|OA| \cdot |OB|)$ 
 * 其中 $OA \cdot OB$ 是向量的数量积 $OA = (Ax - Ox, Ay - Oy)$   $OB = (Bx - Ox, By - Oy)$ ,  $OA \cdot OB = (Ax - Ox) \cdot (Bx - Ox) + (Ay - Oy) \cdot (By - Oy)$ 
 *
 * @param center 顶点 O
 * @param head 点1 A
 * @param touch 点2 B
 * @return
 */
public static float includedAngle(PointF center, PointF head, PointF touch) {
    //  $OA \cdot OB = (Ax - Ox) \cdot (Bx - Ox) + (Ay - Oy) \cdot (By - Oy)$ 
    float AOB = (head.x - center.x) * (touch.x - center.x) + (head.y - center.y) * (touch.y - center.y);
    // OA 的长度
    float OALength = (float) Math.sqrt((head.x - center.x) * (head.x - center.x) + (head.y - center.y) * (head.y - center.y));
    // OB 的长度
    float OBLength = (float) Math.sqrt((touch.x - center.x) * (touch.x - center.x) + (touch.y - center.y) * (touch.y - center.y));
    //  $\cos AOB = (OA \cdot OB) / (|OA| \cdot |OB|)$ 
    float angleCos = AOB / (OALength * OBLength);

    float temAngle = (float) Math.toDegrees(Math.acos(angleCos));
    // 判断方向 正左侧 负右侧 0线上,但是Android的坐标系Y是朝下的,所以左右颠倒一下
    float direction = (center.x - touch.x) * (head.y - touch.y) - (center.y - touch.y) * (head.x - touch.x);
    if (direction == 0) {
        if (AOB >= 0) {
            return 0;
        } else {
            return 180;
        }
    } else {
        if (direction > 0) { // 右侧顺时针为负
            return -temAngle;
        } else {
            return temAngle;
        }
    }
}

```

移动ImageView, 只要给它路径(路径即为上面O,A,C,B几个点求出来的三阶贝塞尔曲线), 通过属性动画可以直接实现。

```
ObjectAnimator objectAnimator = ObjectAnimator.ofFloat(ivFish, "x", "y", path);
```

3-3.鱼身的旋转

鱼转动刚好是贝塞尔曲线的切线。通过PathMeasure可以求出与X轴的角度。

```
objectAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator animation) {
        float fraction = animation.getAnimatedFraction();
        pathMeasure.getPosTan(pathMeasure.getLength() * fraction, null, tan);
        // y轴与实际坐标相反, tan[1] 需要取反
        float angle = (float) (Math.toDegrees(Math.atan2(-tan[1], tan[0])));
        fishDrawable.setFishStartAngle(angle);
    }
});
```