本节目标:

```dart
[1]. 了解如何使用点绘制函数曲线。
[2]. 了解如何根据点进行拟合曲线路径。
[3]. 了解 [图案在路径上的运动] 及 [路径生成动画] 效果。
```

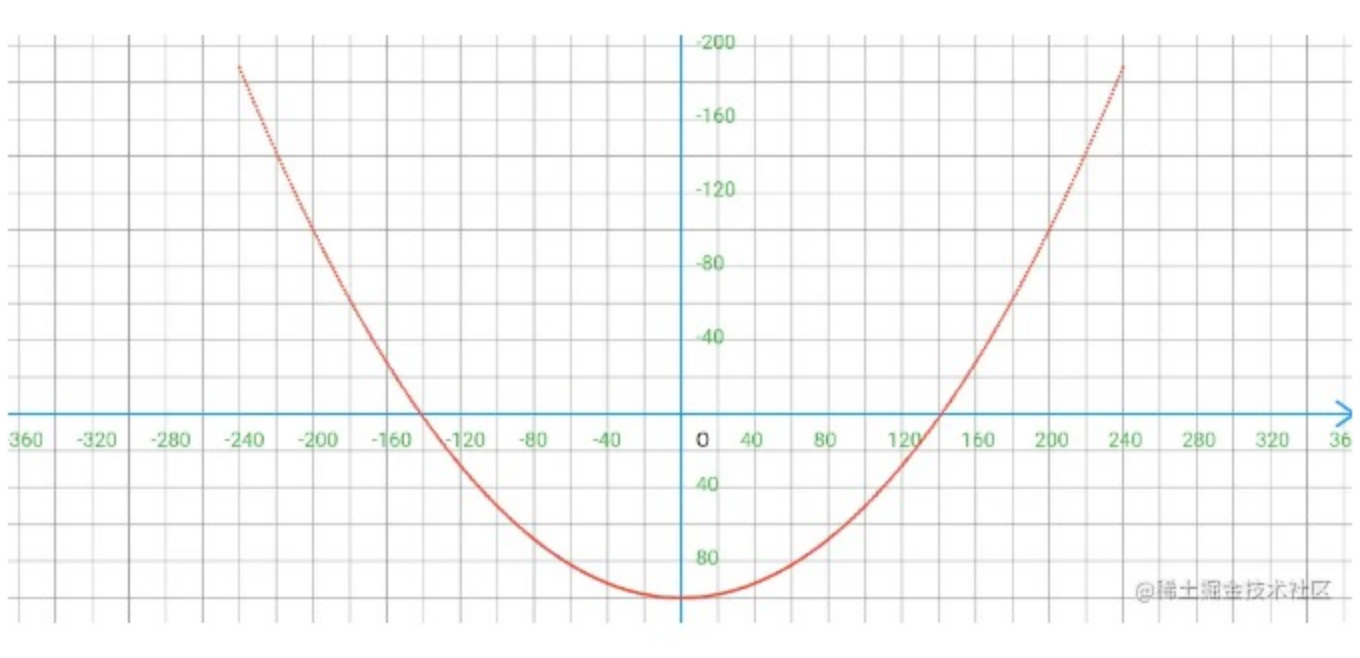
一、使用点线绘制函数曲线

1.如何收集函数曲线的点

> 思路很简单，一个函数比如 `y = - x^2/200 + 100` ，需要画出它在 `[ -240 , 240 ]` 上的曲线，可以遍历区间，根据函数关系计算 y 值即可。

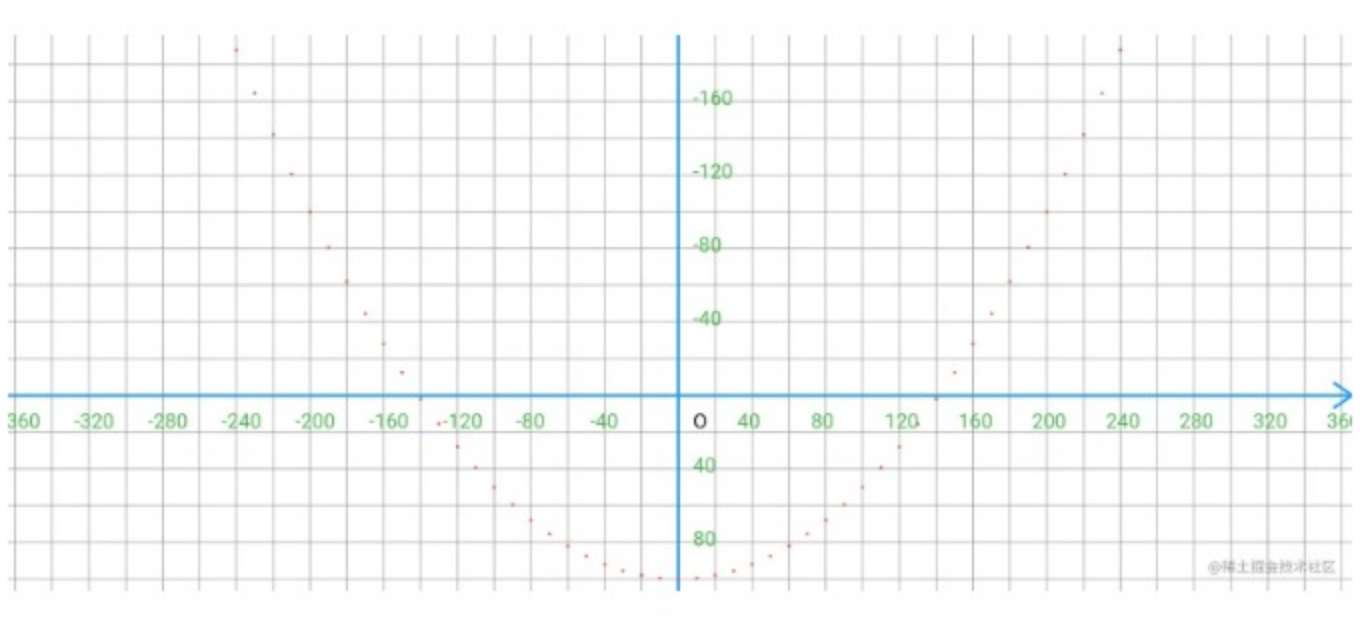
```dart
---->[p13_path_pro/s01_draw_curve_line/paper.dart]----
final List<Offset> points = [];
void initPoints() {
  for (double x = -240; x <= 240; x++) {
    points.add(Offset(x, f(x)));
  }
}
double f(double x) {
  double y = -x * x / 200 + 100;
  return y;
}
```

> 然后绘制出点集， `canvas.drawPoints(PointMode.points, points, paint);`
> 可以看到我们收集到的点是下面的样子。也就是每隔单位 1 记录一点，也就是 480个点。

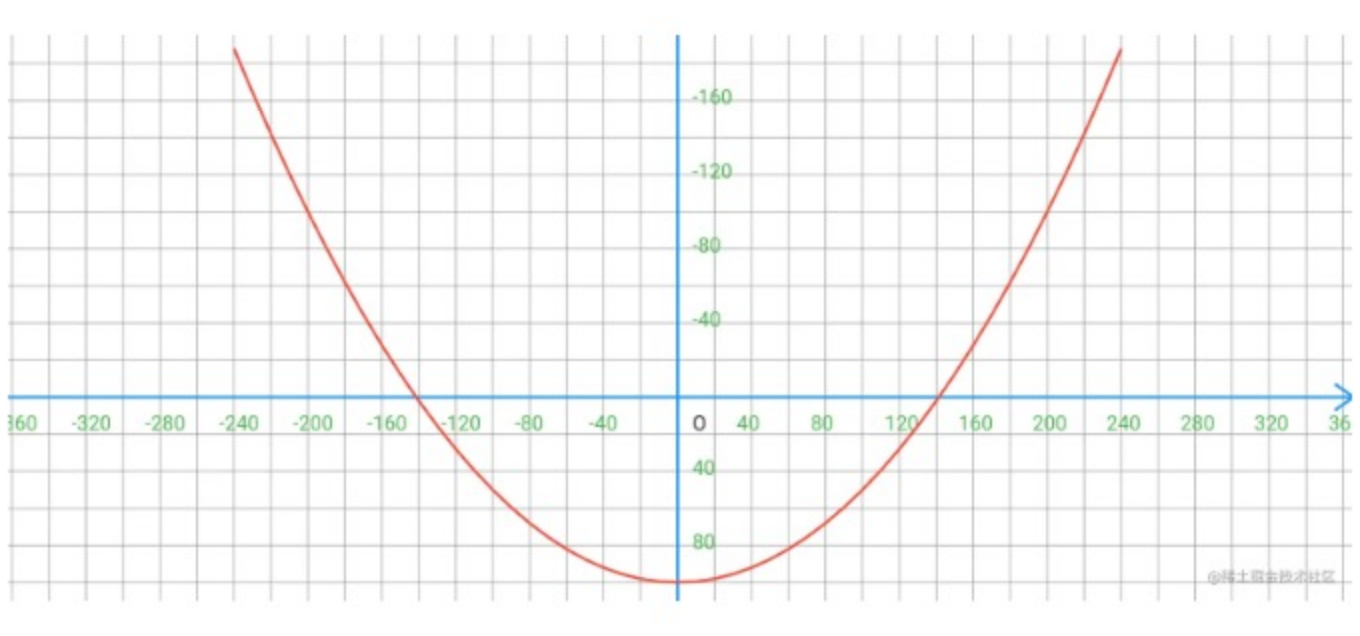


> 我们可以控制自加的进步来减少点的绘制。比如进步为 10 ，那么点数就会少 10 倍

```dart
final double step = 40;
final double min = -240;
final double max = 240;

void initPoints() {
  for (double x = min; x <= max; x += step) {
    points.add(Offset(x,f(x)));
  }
}
```



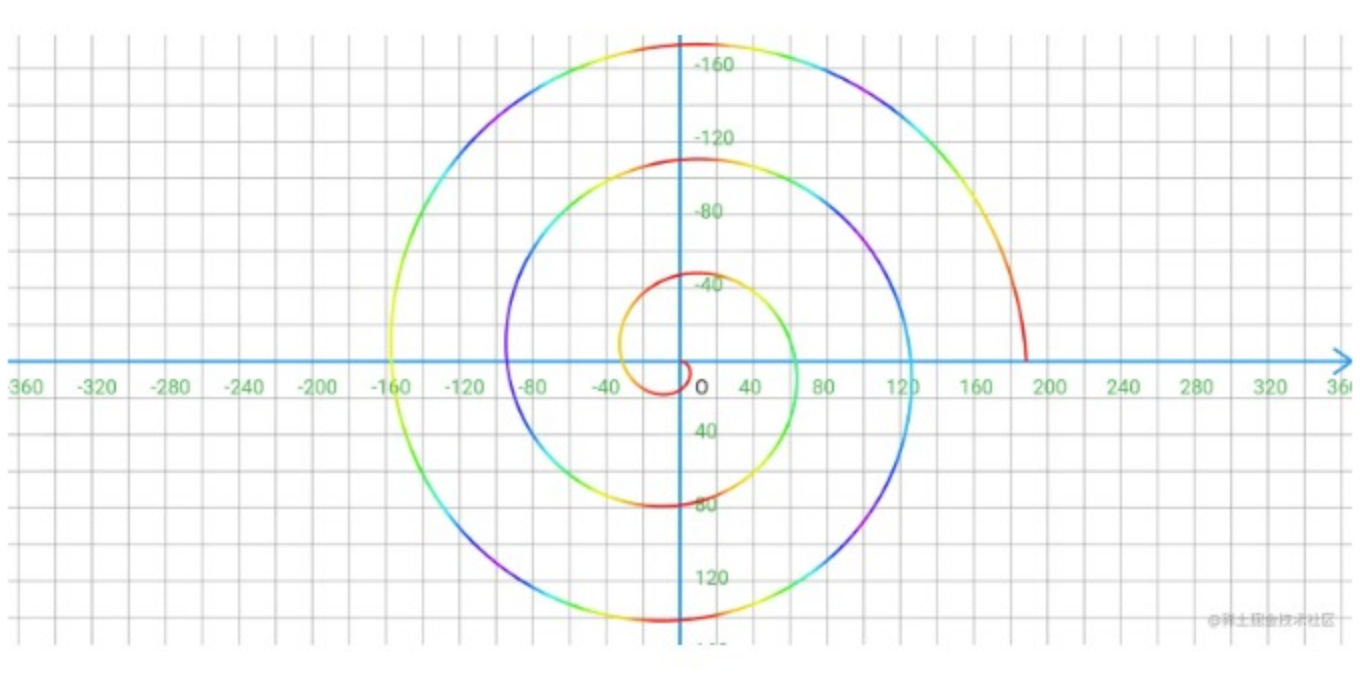
> 将上面的点使用 `PointMode.polygon` 模式绘制，如下，可见效果还是不错的。



---

2. 绘制极坐标的点

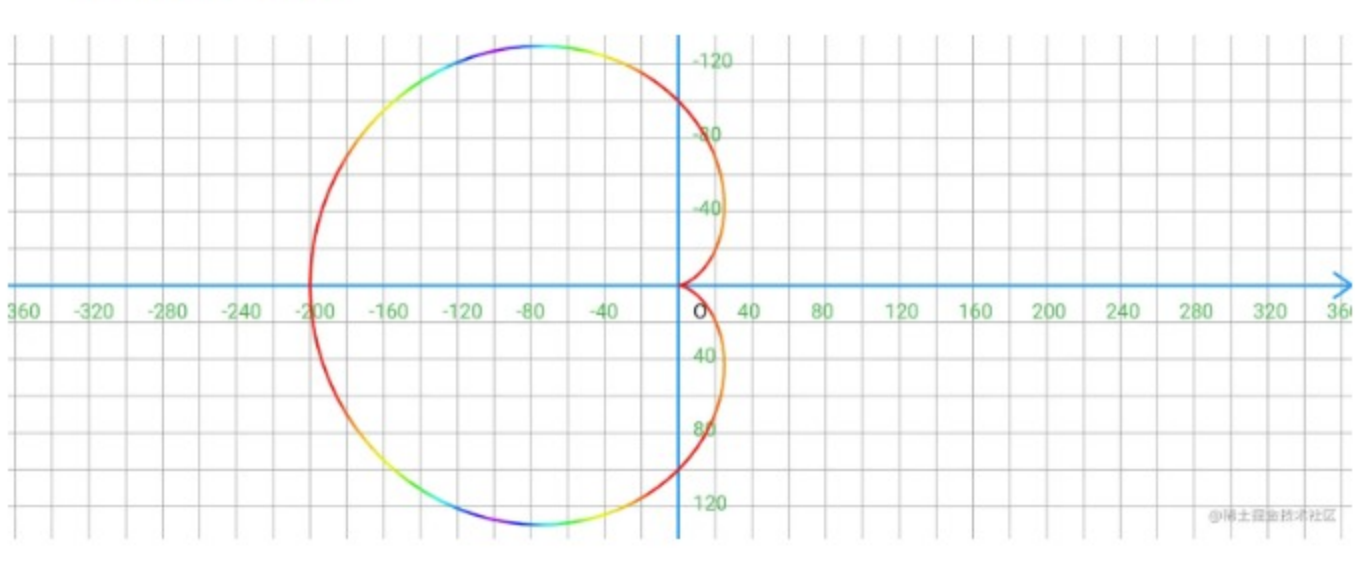
> 极坐标是由 (θ，ρ) 构成的坐标系统，其中 θ 是点与 x 轴的夹角，ρ 是点与原点的长度。
> 极坐标可以很方便地表示出曲线方程，比如下面的螺旋线的表示只是： `ρ = 10 * θ`
> 只要处理好 极坐标和直角坐标 间的转换，其他的就是很简单的事了。



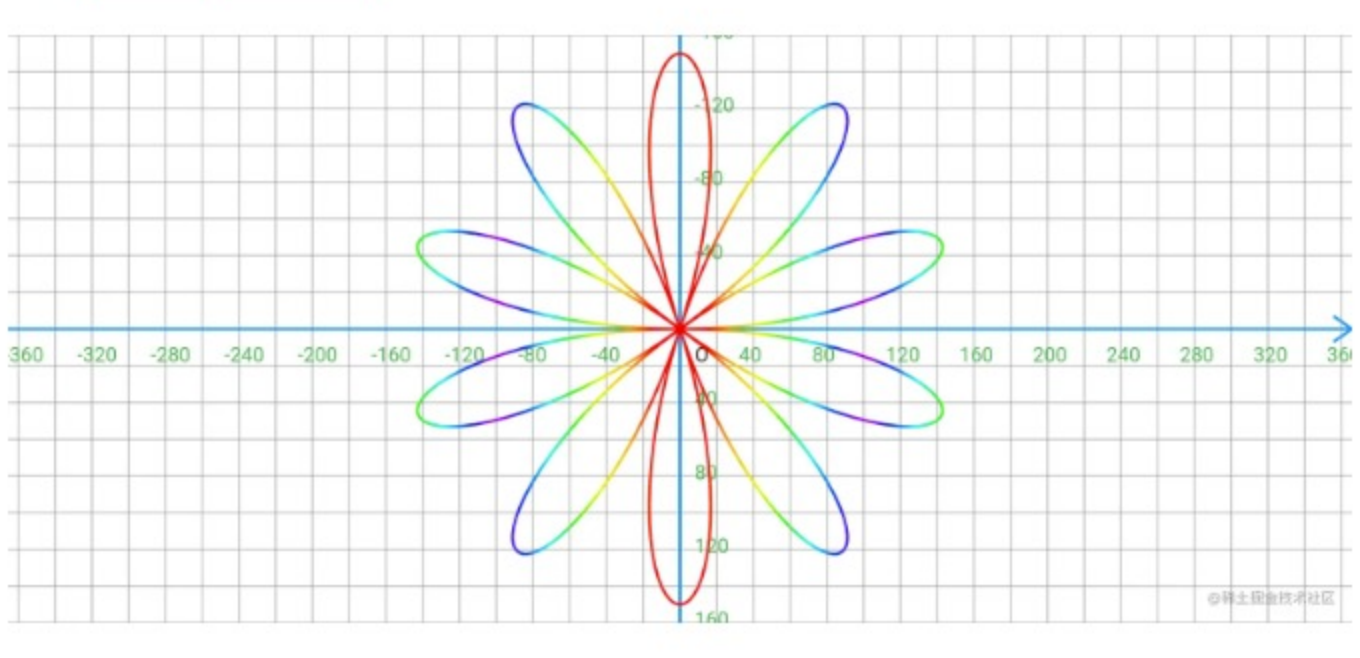
```dart
final double step = 3;
final double min = 0;
final double max = 360*3.0;

void initPointsWithPolar() {
  for (double x = min; x <= max; x += step) {
    double thta = (pi / 180 * x); // 角度转化为弧度
    var p = f(thta);
    points.add(Offset(p * cos(thta), p * sin(thta)));
  }
}

double f(double thta) {
  double p = 10 * thta;
  return p;
}
```

> 你只要替换极坐标方程，就可以画出很多有意思的极坐标图案，下面就列举几个

- `ρ = 100*(1-cosθ)`



- `ρ = 150*|sin(5*θ)|`



- `ρ=(e^(cosθ)- 2cos(4θ) + [sin(θ/12)]^5)*100`

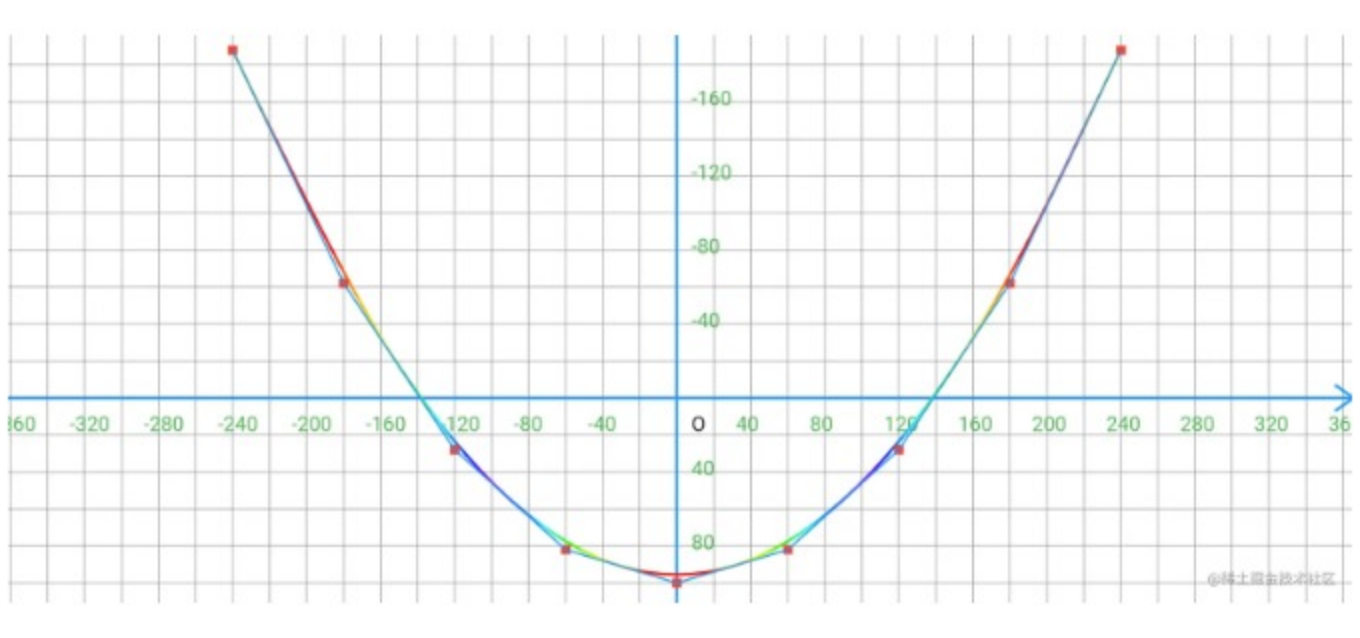
---

3. 存在的问题

> `PointMode.polygon` 是将点依次用直线连接，如下是 `step = 40` 时的样子。可看出进步越小，点越多，曲线就越精细。反之则越粗糙。这样简单的点绘制就出现了它的局限性。
> 另外这样 不能实现 一些 伴随曲线运动 的效果，所以最好还是使用 Path 进行绘制，一方面 Path 可以通过绘制曲线来更好拟合，其次可以通过路径测量完成很多效果。

二、使用 Path 绘制函数曲线

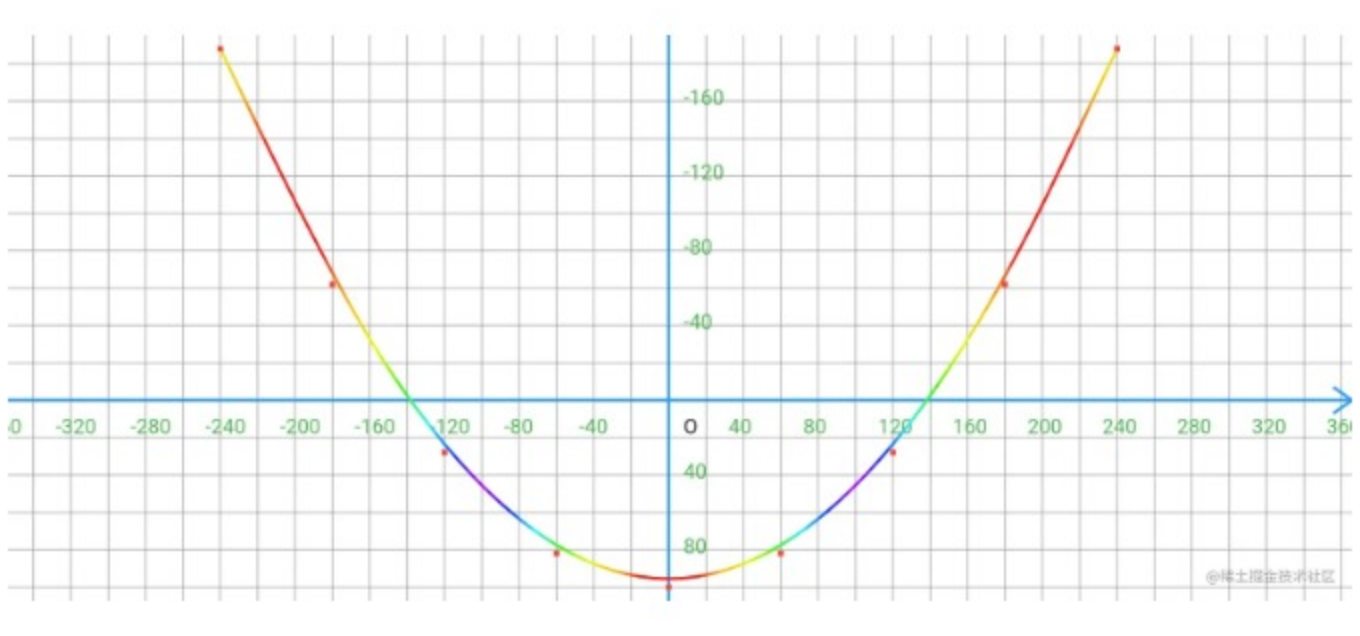
> 下面是 `step = 60` 时的样子，蓝色折线是绘制点的效果，彩色曲线是使用 `Path拟合的曲线` ,可见虽然只有九个点， `Path拟合的曲线` 依然有很好的表现，这就是 Path 的强大之处。下面来看一下如何拟合



> 收录点的方法 `initPoints` ：

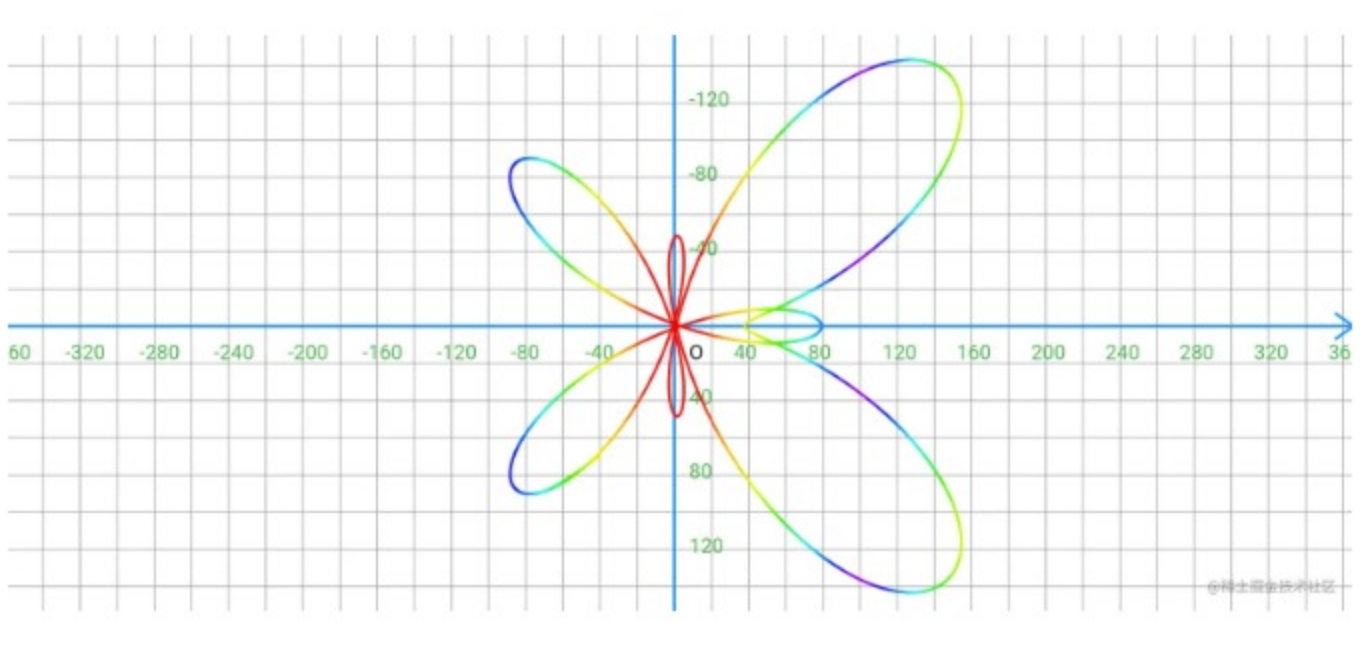
```dart
---->[p13_path_pro/s02_draw_curve_path/paper.dart]----
final double step = 60;
final double min = -240;
final double max = 240;

void initPoints() {
  for (double x = min; x < max; x += step) {
    points.add(Offset(x, f(x)));
  }
  points.add(Offset(max, f(max)));
  points.add(Offset(max, f(max)));
}

double f(double x) {
  double y = -x * x / 200 + 100;
  return y;
}
```

> 使用路径通过两阶贝塞尔曲线拟合(贝塞尔曲线将在后面详细说明)：
> 主要是将除首尾收录的点作为控制点，使用二阶贝塞尔曲线绘制到下两个点中间。



```dart
Offset p1 = points[0];
Path path = Path()..moveTo(p1.dx, p1.dy);
for (var i = 1; i < points.length - 1; i++) {
  double xc = (points[i].dx + points[i + 1].dx) / 2;
  double yc = (points[i].dy + points[i + 1].dy) / 2;
  Offset p2 = points[i];
  path.quadraticBezierTo(p2.dx, p2.dy, xc, yc);
}
```

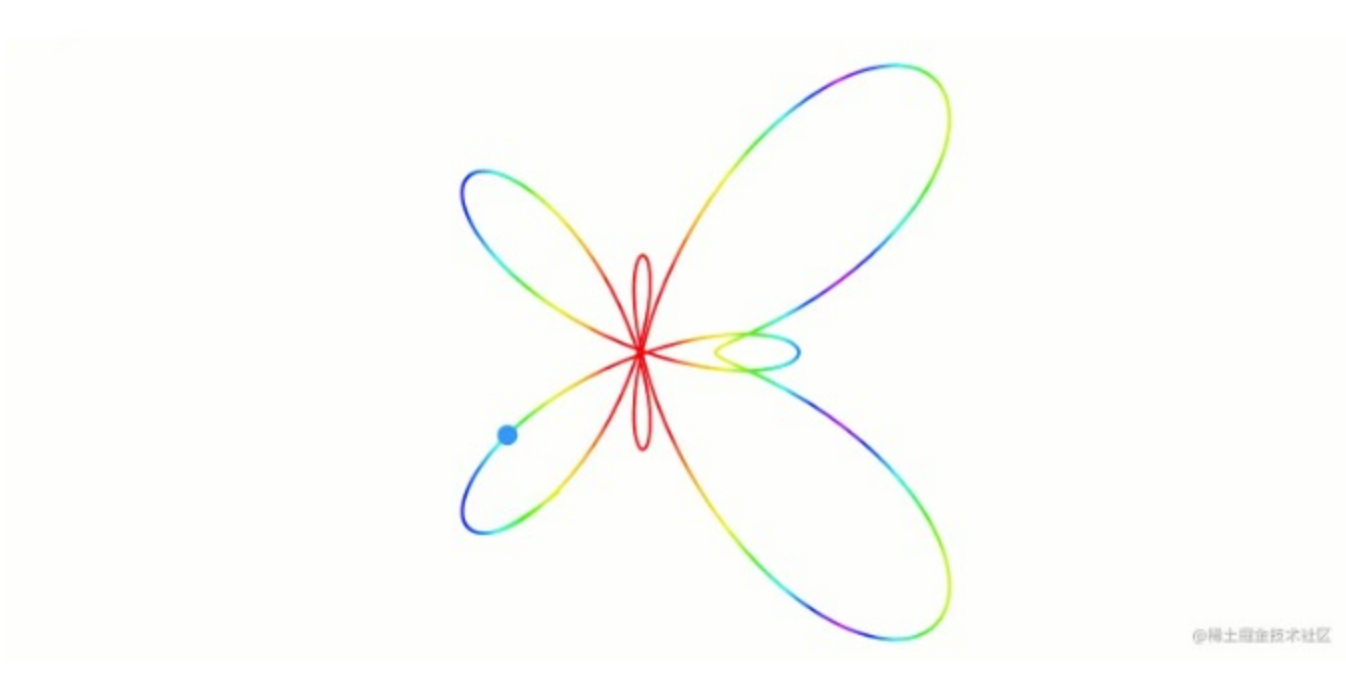
> 使用贝塞尔曲线只是进行拟合，如果点过少也会发生拟合不好的情况。



三、曲线与运动效果

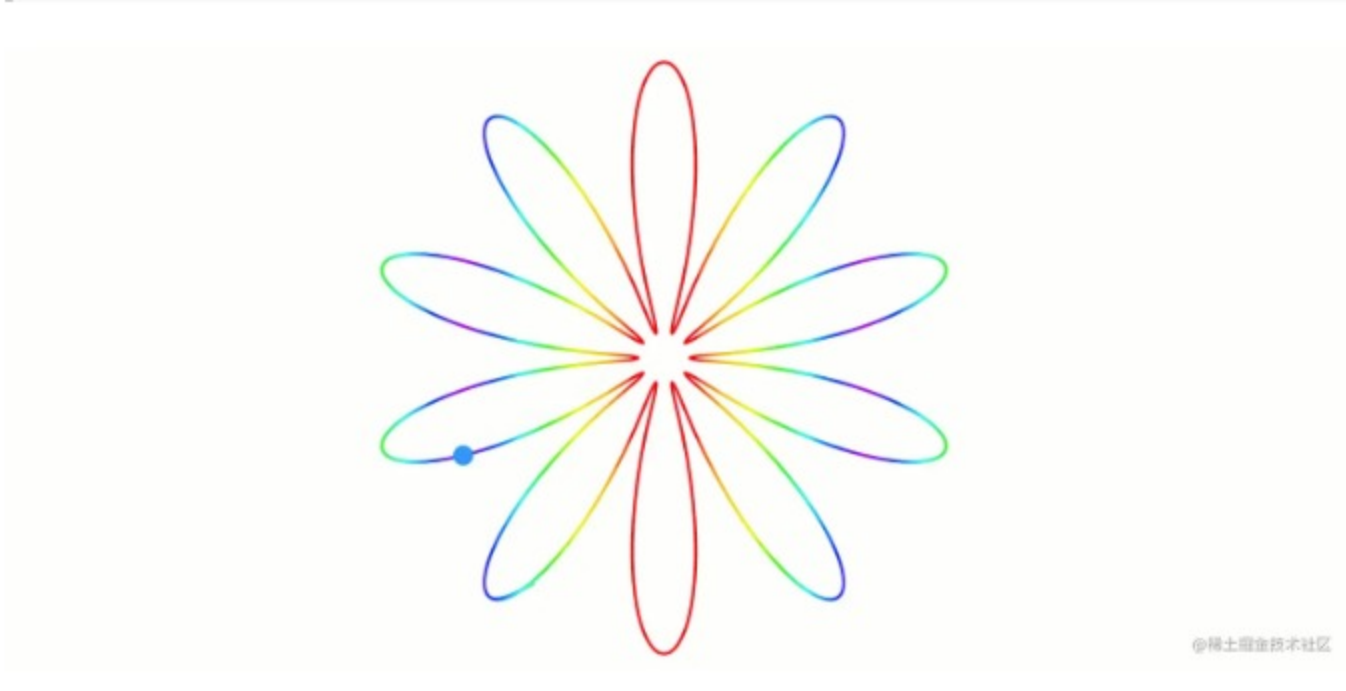
1. 点随函数曲线运动

> 有了 Path ，那么就可以为所欲为了，比如让小球 沿路径运动 ，或让路径产生 绘制过程 的效果。比如下面，小球绕曲线运动。关于这点在 Path 篇的路径测量里说过，主要代码如下。其他代码和前面基本一致，就不贴了，可自己见相应的源码。



```dart
---->[p13_path_pro/s03_curve_path_run/paper.dart]----
PathMetrics pms = path.computeMetrics();
pms.forEach((pm) {
  Tangent tangent = pm.getTangentForOffset(pm.length * repaint.value);
  canvas.drawCircle(
      tangent.position, 5, Paint()..color = Colors.blue);
});
```

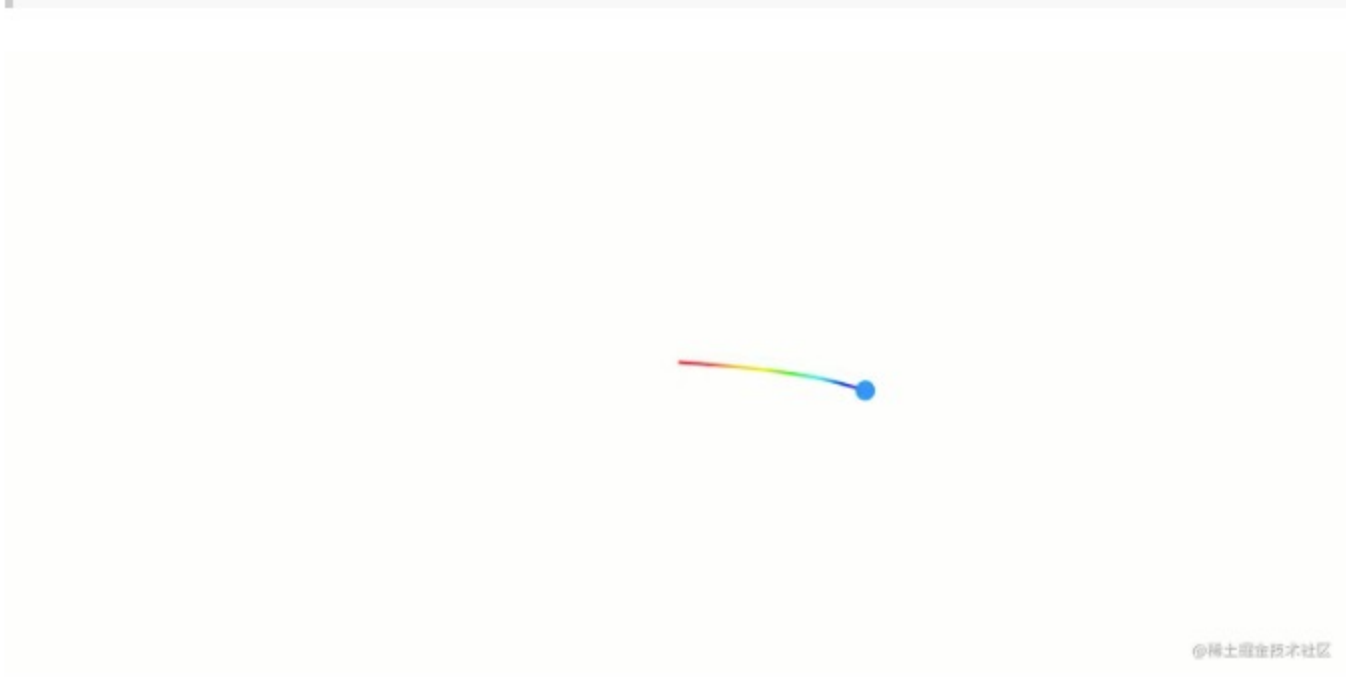
> 只要改变 函数曲线，就可以让小球沿任意曲线运动。



```dart
double f(double thta) {
  double p = 150*sin(5*thta).abs();
  return p;
}
```

2.曲线路径的运动生成

> 路径和动画结合可以做出路径路径的绘制生成，如下所示： 主要方法是 `PathMetric#extractPath` 可以提取出一段长度的路径。



```dart
---->[p13_path_pro/s04_curve_path_run2/paper.dart]----
PathMetrics pms = path.computeMetrics();
pms.forEach((pm) {
  Tangent tangent = pm.getTangentForOffset(pm.length * repaint.value);
  // 提取路径绘制
  canvas.drawPath(pm.extractPath(0, pm.length * repaint.value), paint);
  canvas.drawCircle(
```