

# 使用三次贝塞尔曲线平滑svg路径

翻译

robertCyf

已于 2024-05-07 15:36:42 修改

阅读量507

收藏 1

点赞数 1

分类专栏:

javascript

svg

cubic bezier curves

文章标签:

javascript



javascript 同时被 3 个专栏收录

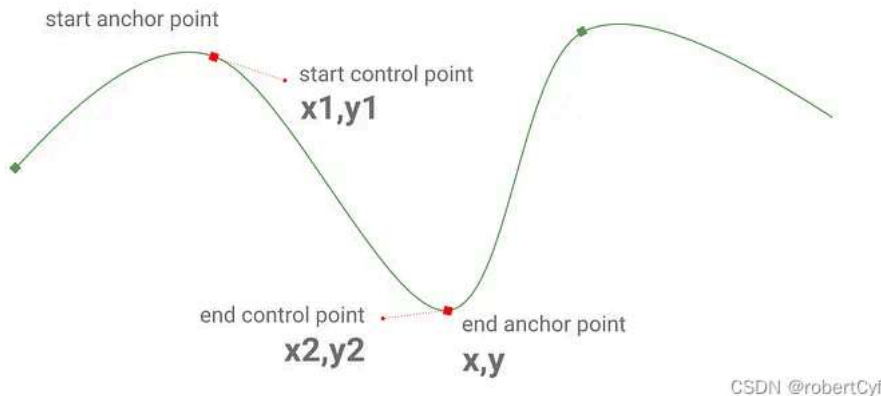
0 订阅 5 篇文章

## 使用三次贝塞尔曲线 平滑svg路径

### 和一点三角 函数

该文章来自于 [smooth-a-svg-path-with-cubic-bezier-curves](#)

虽然在 Svg 元素中绘制直线很简单，但需要一些三角学来平滑这些线。让我们看看如何。



我们有一个 数组 ，表示一条线的点坐标。

```
1 | const points = [[5, 10], [10, 40], [40, 30], [60, 5], [90, 45], [120, 10], [150, 45], [200, 10]]
```

HTML 页面中的 Svg 元素：

```
1 | <svg viewBox="0 0 200 200" version="1.1" xmlns="http://www.w3.org/2000/svg" class="svg"></svg>
```

我们想从点数组中创建一个 <path> 元素。

### 创建path由这个点

<path> 的 d 属性始终以移动到命令： M x,y 开头，后面跟着几个命令，具体取决于形状的类型。结果类似于： <path d="M 10,20 L 15,25 L 2 示直线。

首先，让我们创建一个通用的 svgPath 函数，它有两个参数： point 数组和 command 函数。

```
1 | // Render the svg <path> element
2 | // I: - points (array): points coordinates
3 | //     - command (function)
4 | //     I: - point (array) [x,y]: current point coordinates
5 | //         - i (integer): index of 'point' in the array 'a'
6 | //         - a (array): complete array of points coordinates
7 | //         0: - (string) a svg path command
8 | // 0: - (string): a Svg <path> element
9 | const svgPath = (points, command) => {
10 |   // build the d attributes by looping over the points
11 |   const d = points.reduce((acc, point, i, a) => i === 0
12 |     ? `M ${point[0]},${point[1]}`
13 |     : `${acc} ${command(point, i, a)}`
14 |     , '')
15 |   return `<path d="${d}">
```



robertCyf

关注

1

0

1

0

分享

```
18 | return `<path d="${d}" fill="none" stroke="grey" />`  
   | }
```

现在，让我们创建两个命令函数：

- `lineCommand`：画直线。
- `lineCommand`：画平滑的线。

### 画直线

直线需要指令 `line to`，以字母 `L` 开头，后跟终点 `x, y` 的坐标。

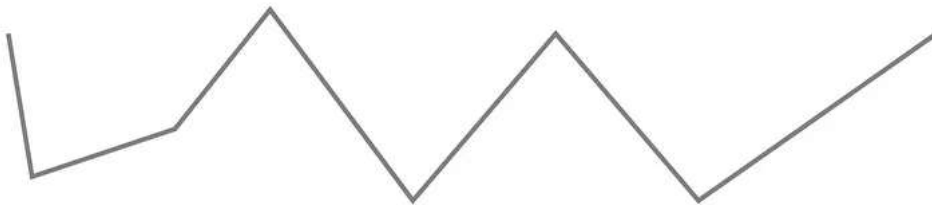
绘制直线的基本 `lineCommand` 函数：

```
1 | // Svg path line command  
2 | // I: - point (array) [x, y]: coordinates  
3 | // O: - (string) 'L x,y': svg line command  
4 | const lineCommand = point => `L ${point[0]} ${point[1]}`
```

现在我们可以用它从点数组中画一条线：

```
1 | const svg = document.querySelector('.svg')  
2 | svg.innerHTML = svgPath(points, lineCommand)
```

这给出了以下结果

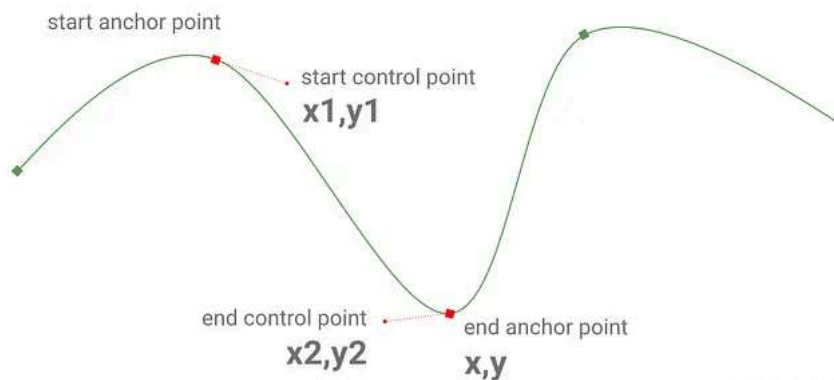


CSDN @robertCyf

### 画平滑的曲线

#### 三次贝塞尔曲线命令

三次贝塞尔曲线命令以字母 `C` 开头，后跟三对坐标 `x1,y1 x2,y2 x,y`：



CSDN @robertCyf

- `x1,y1`：起始控制点坐标
- `x2,y2`：终点控制点坐标
- `x,y`：结束锚点的坐标

需要注意：

- 起始锚点坐标由上一个命令给出。
- 结束锚点坐标来自原始点数组。
- 现在我们必须找到两个控制点的位置。



robertCyf

关注

👍 1

👏

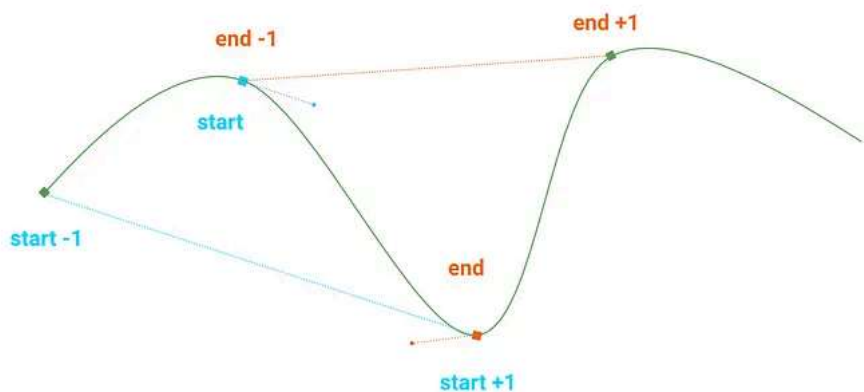
🌟 1

💬 0

🔗 分享

## 找到控制点的位置

我们用一条线将起始 **锚点** 和结束锚点周围的锚点连接起来（我们将其称为 **对立线**）：



CSDN @robertCyf

为了使线条平滑，每个控制点的位置必须相对于其 **对立线**：

- 控制点位于与 **对立线** 平行且与当前锚点相切的线上。
- 在该切线上，从锚点到控制点的距离取决于 **对立线** 的长度和任意 **平滑比**。
- 起始控制点与 **对立线** 的方向相同，而结束控制点则向后。

## code

首先，找到 **对立线** 的属性的函数：

```
1 // Properties of a line
2 // I: - pointA (array) [x,y]: coordinates
3 //     - pointB (array) [x,y]: coordinates
4 // O: - (object) { length: l, angle: a }: properties of the line
5 const line = (pointA, pointB) => {
6   const lengthX = pointB[0] - pointA[0]
7   const lengthY = pointB[1] - pointA[1]
8   return {
9     length: Math.sqrt(Math.pow(lengthX, 2) + Math.pow(lengthY, 2)),
10    angle: Math.atan2(lengthY, lengthX)
11  }
12 }
```

然后，找到控制点位置的函数：

```
1 // Position of a control point
2 // I: - current (array) [x, y]: current point coordinates
3 //     - previous (array) [x, y]: previous point coordinates
4 //     - next (array) [x, y]: next point coordinates
5 //     - reverse (boolean, optional): sets the direction
6 // O: - (array) [x,y]: a tuple of coordinates
7 const controlPoint = (current, previous, next, reverse) => {
8   // When 'current' is the first or last point of the array
9   // 'previous' or 'next' don't exist.
10  // Replace with 'current'
11  const p = previous || current
12  const n = next || current
13  // The smoothing ratio
14  const smoothing = 0.2
15  // Properties of the opposed-line
16  const o = line(p, n)
17  // If is end-control-point, add PI to the angle to go backward
18  const angle = o.angle + (reverse ? Math.PI : 0)
19  const length = o.length * smoothing
20  // The control point position is relative to the current point
21  const x = current[0] + Math.cos(angle) * length
22  const y = current[1] + Math.sin(angle) * length
23  return [x, y]
24 }
```



robertCyf

关注

1



1

0

分享

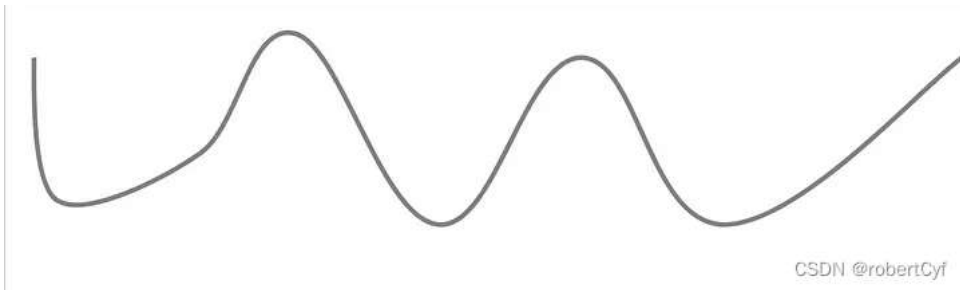
```
    return [x, y]
  }
```

创建贝塞尔曲线的函数 **C** 命令:

```
1 // Create the bezier curve command
2 // I: - point (array) [x,y]: current point coordinates
3 //     - i (integer): index of 'point' in the array 'a'
4 //     - a (array): complete array of points coordinates
5 // 0: - (string) 'C x2,y2 x1,y1 x,y': SVG cubic bezier C command
6 const bezierCommand = (point, i, a) => {
7   // start control point
8   const [cpsX, cpsY] = controlPoint(a[i - 1], a[i - 2], point)
9   // end control point
10  const [cpeX, cpeY] = controlPoint(point, a[i - 1], a[i + 1], true)
11  return `C ${cpsX},${cpsY} ${cpeX},${cpeY} ${point[0]},${point[1]}`
12 }
```

最后, 我们重用 **svgPath** 函数来循环数组的点并构建 **<path>** 元素。然后将 **<path>** 附加到 **<svg>** 元素。

```
1 const svg = document.querySelector('.svg')
2 svg.innerHTML = svgPath(points, bezierCommand)
```



CSDN @robertCyf

#### SVG Path (四) 贝塞尔曲线命令

卡尔4

贝塞尔曲线示例, 分别为 1、2、3、4 次贝塞尔曲线。二次贝塞尔曲线: 三次贝塞尔曲线: M x0 y0 C x1 y1 x2 y2 x y (x0, y0): 起始点 (x1, y1): 起始控制点 (x2, y2): 结

#### 无人驾驶全局路径规划之路径平滑 (贝塞尔曲线)

qq\_43301351的博

无人驾驶, 全局路径规划, 路径平滑, 贝塞尔曲线

#### SVG中的d属性(贝塞尔曲线)

SVG定义了6种路径命令,总共20条命令: 搬去:M,m 线路:L,l,H,h,V,v 三次贝塞尔曲线:C,c,S,s 二次贝塞尔曲线:Q,q,T,t 椭圆弧曲线:A,a 关闭路径:Z,z 注意:命令区分大小写。大

#### MyPrint打印设计器(八)svg篇-三阶贝塞尔曲线\_js svg 贝塞尔曲线

首先,我们需要在HTML中设置基本的SVG元素,以便绘制图形。<!DOCTYPEhtml>MyPrint|打印设计|SVG|三阶贝塞尔曲线body{width:100vw;height:100vh;margin:0;display

#### ROS导航使用贝塞尔曲线对全局路径进行平滑处理 最新发布

weixin\_44126988的博

ROS原生的全局路径规划GlobalPlanner包含A\*和Dijkstra, 两者原理基本相同, 能够规划出从起点到终点的路径, 但是由于栅格地图存在锯齿形, 得到的全局路径也会出现

#### svg中画三次贝塞尔曲线

weixin\_67544162的博

传入点数组, 点对象有x, y值。

#### SVG路径详解:移动、贝塞尔曲线与弧线绘制

</svg> 1 2 3 4 3) Q二次贝塞尔曲线 两次贝塞尔曲线有一个中间点,quadratic是两次的,二次方程式的意思,所以使用Q命令来进行二次贝塞尔曲线绘制。<path d=

#### svg path 详解

</svg> 原理分析:结合下面的图看一下,曲线沿着起点到第一控制点的方向伸出,逐渐弯曲,然后沿着第二控制点到终点的方向结束。S(smooth curveto)简写的三次贝塞尔曲线

#### 深度掌握SVG路径path的贝塞尔曲线指令

SalmonellaVaccine的博

by zhangxinxu from http://www.zhangxinxu.com 本文地址: http://www.zhangxinxu.com/wordpress/?p=4197 一、数字、公式、函数、变量, 哦, NO! 又一次说起贝塞尔

#### 【路径规划】贝塞尔曲线平滑路径

龙猫的博

贝塞尔曲线一般是用于二维图形的一种数学曲线, 一般是用于一些矢量图的设计, 不过在路径规划中, 也可以应用上, 例如之前的RRT随机搜索算法, 因为是随机搜索, 且

#### SVG高级技巧详解

前面分享过一篇基础svg的,这篇分享一些高级的SVG教程,比如SVG路径中的三次贝塞尔曲线、椭圆弧线、SVG文本和填充效果等。一.SVG路径 (一)直线命令 path元素里:

#### 深度掌握SVG路径path的贝塞尔曲线指令\_svg 曲线方程

深度掌握SVG路径path的贝塞尔曲线指令 byzhangxinxufromhtt



robertCyf

关注

1



1



0

分享