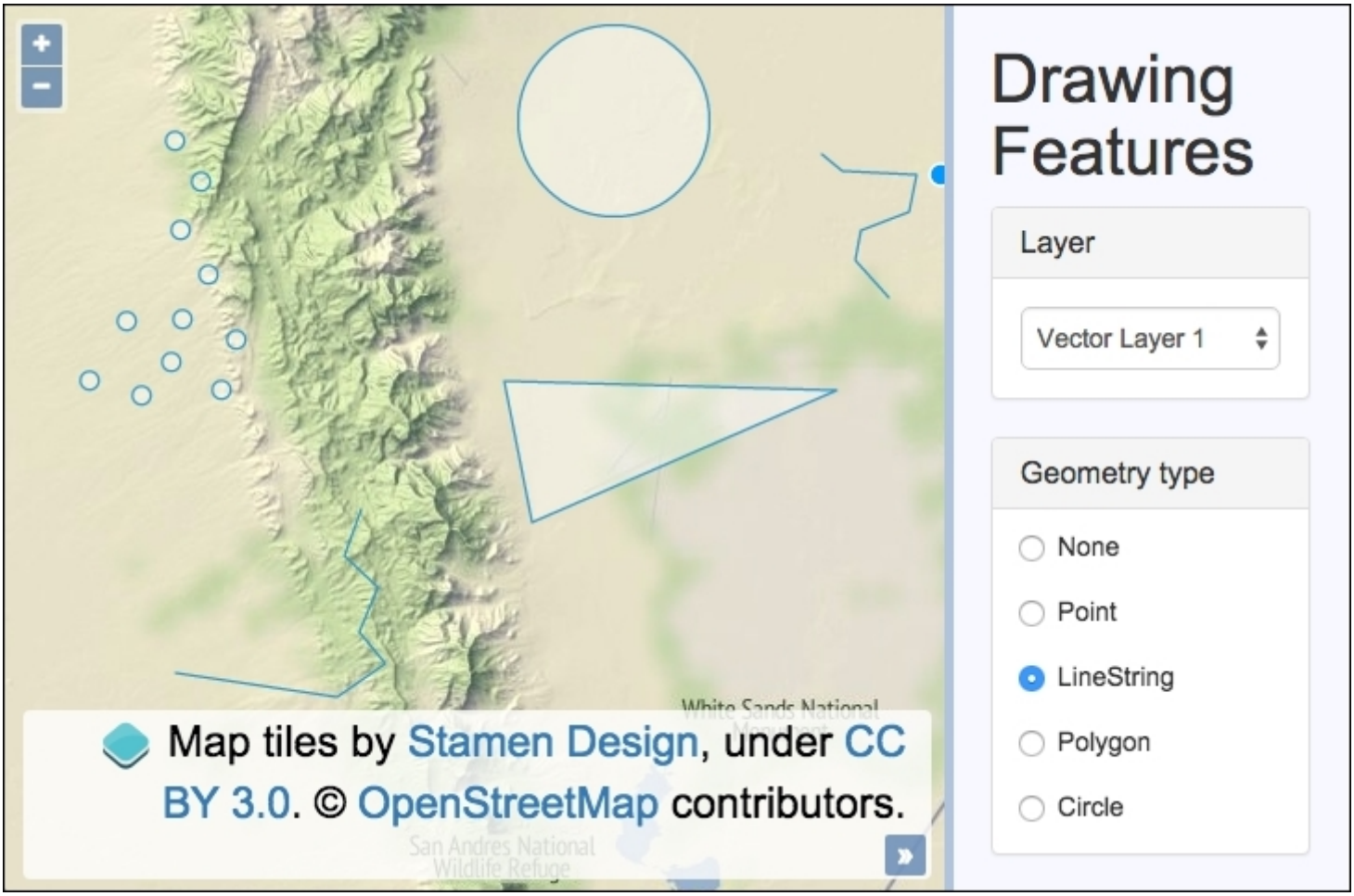
# 跨多个矢量层绘制要素

我们已经看到了多种方法 向地图添加矢量要素，例如作为配方，**添加GML层**从这一章，并**以编程方式创建功能**在第3章 (/book/web_development/9781785287756/3)，**工作与矢量图层。**但是，有时我们可能希望为用户提供绘图功能，以便他们可以在地图上手动绘制几乎任何喜欢的形状。

例如，您可能希望用户通过在区域上绘制多边形以进行建筑规划来在某个位置标记出兴趣点。

OpenLayers有很多控件（更准确地说是交互），我们可以为这种目的添加到地图中。

在本食谱中，我们将允许用户从不同的几何类型（**Point**，**LineString**，**Polygon**和**Circle**）中进行选择，并将其添加到他们选择的矢量层中。可以在中找到源代码 `ch05/ch05-drawing-features` ，其外观如下所示：




怎么做...

了解如何绘制 不同的矢量要素类型 请按照以下说明在多个矢量层上进行操作：

1 创建具有OpenLayers依赖项，jQuery的HTML文件，并 `div` 保存地图。特别是，我们需要一个 `select` 菜单，其中包含可用的矢量层以及 `radio` 按钮，以便可以选择几何类型：

复制

```
<select id="js-layer-select">
<option value="vectorLayer1">Vector Layer 1</option>
<option value="vectorLayer2">Vector Layer 2</option>
</select>
<label>
<input type="radio" name="geometries" value="None" checked>
  None
</label>
<label>
<input type="radio" name="geometries" value="Point">
  Point
</label>
<label>
<input type="radio" name="geometries" value="LineString">
  LineString
</label>
<label>
<input type="radio" name="geometries" value="Polygon">
  Polygon
```

2 创建一个 自定义JavaScript 文件并设置两个矢量层：

复制

```
var vectorLayer1 = new ol.layer.Vector({
  source: new ol.source.Vector()
});
var vectorLayer2 = new ol.layer.Vector({
  source: new ol.source.Vector()
});
```

3 `map` 用 `view` 实例和图块层实例化并添加矢量层：

复制

```
var map = new ol.Map({
  view: new ol.View({
    zoom: 10, center: [-11863791, 3898899]
  }),
  target: 'js-map',
  layers: [
    new ol.layer.Tile({ source: new ol.source.Stamen({
      layer: 'terrain'
    })}), vectorLayer1, vectorLayer2
  ]
});
```

4 缓存一些DOM元素并设置一个变量来保存绘图控件：

复制

```
var layerSelect = $('#js-layer-select');
var geomRadios = $('[name=geometries]');
var drawControl;
```

5 创建 可以的功能 相应地更新绘图控件：

复制

```
var updateDrawControl = function() {
  var geometryType = geomRadios.filter(':checked').val();
  map.removeInteraction(drawControl);

  if (geometryType === 'None') return;

  drawControl = new ol.interaction.Draw({
    type: geometryType,
    source: window[layerSelect.val()].getSource()
  });
  map.addInteraction(drawControl);
};
```

6 最后， `change` 在 `select` 菜单和 `radio` 输入上都订阅该事件，并将 `updateDrawControl` 函数设置为两者的处理程序：

复制

```
layerSelect.on('change', updateDrawControl);
geomRadios.on('change', updateDrawControl);
```

## 怎么运行的...

正如我们在其他食谱中所做的那样，CSS框架Bootstrap已用于设置边栏内容的样式。要查看完整的HTML和CSS，请查看源代码，因为它太大了，无法包含在本书中。

本部分的重点将放在我们创建的函数上，即 `updateDrawControl` ，这是逻辑的核心所在。这也是您到目前为止所看到的其他食谱最不熟悉的地方。

让我们开始消化函数的每一部分，每当从 `select` 菜单更改矢量层或从 `radio` 输入更改几何类型时，都会调用该函数：

复制

```
var updateDrawControl = function() {
  var geometryType = geomRadios.filter(':checked').val();
```

我们将用户从 `radio` 输入列表中选择的几何类型缓存到变量 `geometryType` 。这是借助于jQuery的方法来实现的，该方法 `filter` 遍历集合（ `radio` 输入元素的列表）并确定是否检查了输入（由于 `:checked` 传入的条件）。

如果 `radio` 检查了迭代期间的输入，则该 `filter` 方法从集合中返回匹配的DOM元素，我们通过该 `val` 方法从集合中检索输入值，该方法也来自jQuery。例如，这可以等于“ **Circle** ”，“ **LineString** ”或用户可能选择的任何其他选项。

复制

```
map.removeInteraction(drawControl);
if (geometryType === 'None') return;
```

本节 该功能首先删除任何 `interaction` 地图中绘图（控件）的先前实例。这样做是因为必须 `interaction` 从 `select` 菜单（图层）和 `radio` 输入（几何类型）的组合中实例化最新设置。在OpenLayers API的限制内，如果不破坏然后重新创建交互，我们就无法更新几何类型或图层。

然后，我们有一个条件 `if` 语句来查看用户是否从几何类型中选择了“ **无** ”。这意味着用户对绘图不感兴趣，而是打算浏览地图。从地图上删除绘图交互后，这是我们在此情况下需要做的所有事情，因此我们通过尽早返回，完成工作来打破功能。

复制

```
drawControl = new ol.interaction.Draw({
  type: geometryType,
  source: window[layerSelect.val()].getSource()
});
map.addInteraction(drawControl);
```

我们将绘图交互的新实例分配给变量 `drawControl` 。这一点很重要，因为 `interaction` 当我们从地图中删除它时我们需要一个引用（如本功能前面所述）。

用户要绘制的几何类型在 `type` 属性上设置。的 `source` 属性（矢量源）需要的我们的双载体层来源之一。向量层是全局变量，因此它们是 `window` 对象的属性。 `select` 选项的值等于矢量层变量的名称。这意味着，当选择**Vector Layer 1**时，将通过jQuery 方法 `vectorLayer1` 检索（记住，HTML看起来像这样 `<option value="vectorLayer1">Vector Layer 1</option>` ：）的值 `val` 。通过在 `window` 对象上进行数组符号查找，我们可以引用矢量层。例如， `window[layerSelect.val()]` 可能等于 `window[vectorLayer1]` 。

我们 `getSource` 从向量层调用该方法，这就是提供给 `source` 属性的内容。

新实例化的绘图交互已添加到中 `map` ，因此，用户现在可以在地图上绘制特征。

不一定很明显 如何完成一个图纸 功能，所以有一个细分：

- **点**：这是单击地图绘制一个点的时间。
- **LineString / Polygon**：这是在最初的单击添加起点并且随后的单击绘制出特征的新点时。双击完成几何。

- **圆**：这是指在地图上单击以设置圆的中心，然后向外拖动鼠标直到对直径满意为止，然后再次单击以完成绘图。对于移动设备，第二次单击所需的半径即可完成该圆。

本 `ol.interaction.Draw` 类有实例化许多配置属性，可能会感兴趣，如 `minPoints` 和 `maxPoints` ，限制的几何点，该功能可以有量。还有一个 `features` 属性，它将是绘制要素的目标要素集合。我们还需要看看 `freehandCondition` 在配方属性，**在绘制写意模式**，在第7章(/book/web_development/9781785287756/7)，**超越基础。**

该 `ol.interaction.Draw` 班还发布了，你可能想订阅的事件，如自我解释 `drawend` 和 `drawstart` 事件。

## 还有更多...

OpenLayers通过绘图交互提供了开箱即用的设置几何类型，还有更先进的技术可用于绘制几乎任何您需要的形状。

绘图交互具有名为属性 `geometryFunction` ，该属性接受type `ol.interaction.DrawGeometryFunctionType` 函数的功能。此函数将坐标和现有几何作为参数传递，并且它必须返回新的几何。

使用此方法，您可以操纵诸如圆形几何类型之类的绘图机制，而是通过从单击点向外拖动鼠标来绘制正方形。该 `ol.interaction.Draw` 级配备了一个画法几何方法，你可以轻松地定制叫 `ol.interaction.Draw.createRegularPolygon` ，这需要一个边数和角度作为参数。为了演示绘制正方形（使用与绘制圆形相同的手势动作），可以使用以下内容：

复制

```
drawControl = new ol.interaction.Draw({
  type: 'Circle',
  source: vectorLayer1.getSource(),
  geometryFunction: ol.interaction.Draw.createRegularPolygon(4)
});
```

这是一个高级主题 不幸的是不会被覆盖本书会提供更多详细信息，但有关更多信息，请访问OpenLayers的官方示例，网址为http://openlayers.org/en/v3.13.1/examples/draw-features.html (http://openlayers.org/en/v3.13.1/examples/draw-features.html)。

## 也可以看看

- **地图配方外的“ 放置”控件**
- **该改性特征食谱**

上一节 (/book/web_development/9781785287756/5/ch05lvl1sec48/placing-controls-outside-th

下一节 (/book/web_development/9781785287756/5/ch05lvl1sec50/modifying-features)