实验 6：Canis/Cast/Libra 实践

大数据分析实践实验报告

姓　　名: 郑坤武

学　　号: 202200150184

班　　级: 22 级公信班

学　　院: 政治学与公共管理学院

2025 年 11 月 27 日

1 实验目的

本实验旨在基于 **Canis: A High-level Language for Data-Driven Chart Animations**（EuroVis 2020）的思想，模拟实现一个轻量级的 **动画编译器**，使得用户能够通过 **声明式 JSON 配置**来生成数据驱动的动画。实验重点包括：

- 理解并实现 **dSVG**（Data-enriched SVG）概念，并用其模拟数据可视化。
- 掌握 **Canis 声明式语法**（Spec），通过 JSON 配置来描述动画。
- 模拟 Canis 中的编译器工作流，完成动画的生成与展示。
- 使用简单的 **JavaScript 引擎**解析 JSON 配置并驱动动画效果。

2 实验环境

- **操作系统：** macOS 12.x
- **开发工具：** VS Code + 终端 (Terminal)
- **语言与运行环境：** HTML + JavaScript (浏览器环境)
- **主要依赖库：** 无
- **浏览器：** Google Chrome 或 Safari（用于运行 HTML 文件）

3 具体实验步骤与结果分析

3.1 项目初始化与环境配置

本实验不依赖于特定的框架或库，而是通过纯 HTML 与 JavaScript 来模拟 Canis 的编译器。创建 index.html 文件, 将以下代码复制到该文件中，并在浏览器中打开以进行测试和调试。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <title>Canis Grammar Simulator</title>
    <style>
        body { font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; padding: 40px;
            background: #f4f4f4; }
        .container { display: flex; gap: 20px; }
        .panel { background: white; padding: 20px; border-radius: 8px; box-shadow: 0 2px 5px rgba
            (0,0,0,0.1); }
        #chart-area { width: 400px; height: 400px; border: 1px solid #eee; }
        textarea { width: 300px; height: 300px; font-family: monospace; font-size: 14px; border: 1px
             solid #ddd; background: #2d2d2d; color: #ccc; padding: 10px;}
        h3 { margin-top: 0; color: #333; }
        button { background: #4a90e2; color: white; border: none; padding: 10px 20px; border-radius:
             4px; cursor: pointer; margin-top: 10px; }
        button:hover { background: #357abd; }
```

```
15
16          /* 基础图元样式 */
17          circle { transition:  all 0.5s ease; opacity: 0; }
18      </style>
19  </head>
20  <body>
21
22      <h2>Canis 实践: 声明式动画编译器模拟</h2>
23      <p>参考: <cite>Canis: A High-level Language for Data-Driven Chart Animations (EuroVis 2020)</
            cite></p>
24
25      <div class="container">
26          <div class="panel">
27              <h3>1. Output Chart (dSVG Render)</h3>
28              <svg  id="chart-area" viewBox="0 0 400 400"></svg>
29          </div>
30
31          <div class="panel">
32              <h3>2. Canis Specification (JSON)</h3>
33              <p>修改下方的配置来控制动画逻辑: </p>
34              <textarea  id="canis-spec">
35  {
36    "selector": "circle",
37    "grouping": "color",
38    "animation": {
39      "type": "fade",
40      "duration": 800,
41      "delayBetweenGroups": 500
42    }
43  }
44              </textarea>
45              <br>
46              <button onclick="runCompiler()">Run Compiler & Animate</button>
47          </div>
48      </div>
49
50      <script>
51          // === 步骤 1: 模拟 dSVG 数据输入 ===
52          const rawData = [];
53          const categories = ['A', 'B', 'C'];
54          const colors = { 'A': '#ff6b6b', 'B': '#4ecdc4', 'C': '#feca57' };
55
56          for(let i=0; i<60; i++) {
57              const cat = categories[i % 3];
58              rawData.push({
59                   id: i,
60                  category: cat,
61                  x: Math.random() * 300 + 50,
62                  y: Math.random() * 300 + 50,
```

```
            color: colors[cat]
        });
    }

    const svg = document.getElementById('chart-area');

    // 渲染静态 dSVG (初始状态不可见)
    function initDSVG() {
        svg.innerHTML = '';
        rawData.forEach(d => {
            const circle = document.createElementNS("http://www.w3.org/2000/svg", "circle");
            circle.setAttribute("cx", d.x);
            circle.setAttribute("cy", d.y);
            circle.setAttribute("r", 6);
            circle.setAttribute("fill", d.color);
            circle.setAttribute("data-category", d.category);
            circle.setAttribute("class", "mark-unit");
            circle.style.opacity = 0;
            circle.style.transform = "scale(0)";
            svg.appendChild(circle);
        });
    }

    function runCompiler() {
        initDSVG();
        let spec;
        try {
            spec = JSON.parse(document.getElementById('canis-spec').value);
        } catch(e) { alert("JSON 格式错误"); return; }

        const circles = Array.from(document.querySelectorAll('.mark-unit'));
        let groups = {};

        if (spec.grouping === 'color') {
            circles.forEach(circle => {
                const key = circle.getAttribute('data-category');
                if (!groups[key]) groups[key] = [];
                groups[key].push(circle);
            });
        } else {
            groups['all'] = circles;
        }

        const groupKeys = Object.keys(groups);
        groupKeys.forEach((key, groupIndex) => {
            const elements = groups[key];
            const startTime = groupIndex * spec.animation.delayBetweenGroups;

            elements.forEach((el, elIndex) => {
```

```
                    setTimeout(() => {
                        applyAnimation(el, spec.animation. type, spec.animation.duration);
                    }, startTime + (elIndex * 20));
                });
            });
        }

        function applyAnimation(element,  type, duration) {
            element.style.transition = ` all ${duration}ms cubic-bezier(0.25, 0.8, 0.25, 1)`;
            element.style.opacity = 1;

            if ( type === 'fade') {
                element.style.transform = "scale(1)";
            }
        }

        initDSVG();
    </script>
</body>
</html>
```

3.2 实验步骤与结果分析

初始化数据：模拟了一个包含三个类别（A、B、C）的数据集，每个类别包含 20 个数据点。每个数据点都通过一个带有数据属性（如 category 和 color）的 SVG 圆点进行表示。

声明式配置 (Spec)：通过修改右侧的 JSON 配置，用户可以控制动画的分组（如按颜色分组）以及动画的类型（如渐显）和持续时间。

动画生成：用户点击”Run Compiler & Animate” 按钮后，页面根据配置自动生成动画。根据 grouping 设置的不同，动画的效果会有所不同。

3.2.1 修改参数前后效果对比

通过修改 JSON 配置文件中的参数，可以观察到不同的动画效果：

修改前：默认配置为 grouping: ”color”，动画按颜色分组依次出现（颜色 A → B → C），每个组之间有 500 毫秒的延迟。

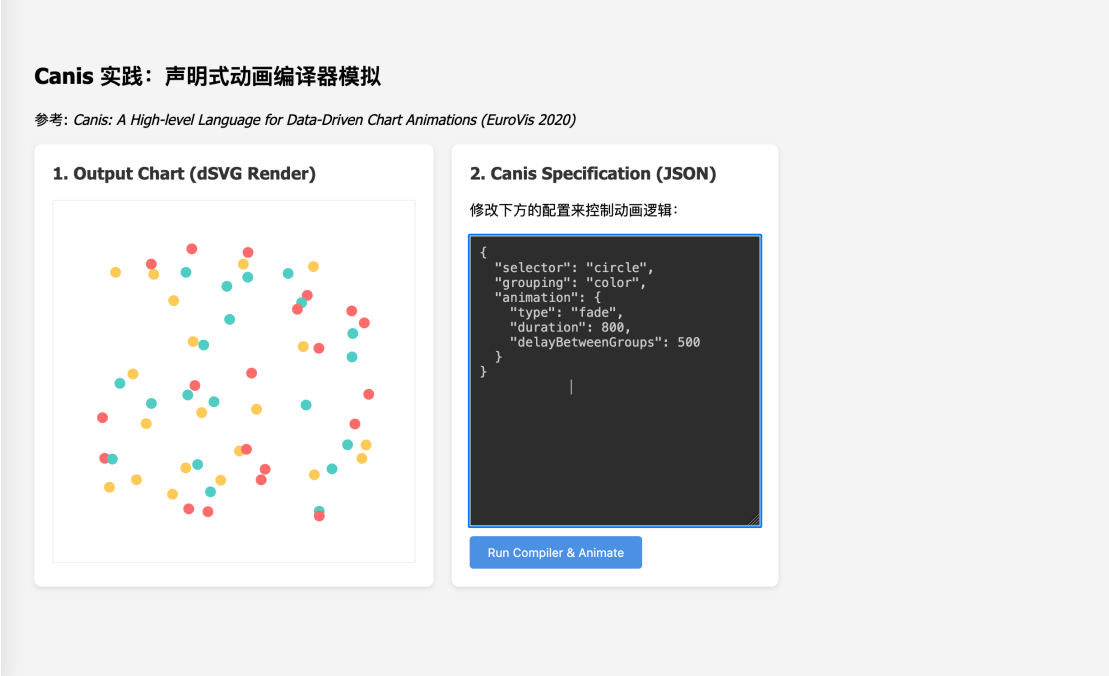

图 1: 修改前

修改后：如果将 grouping: ”none”，所有点会同时出现，不再按颜色分组展示。

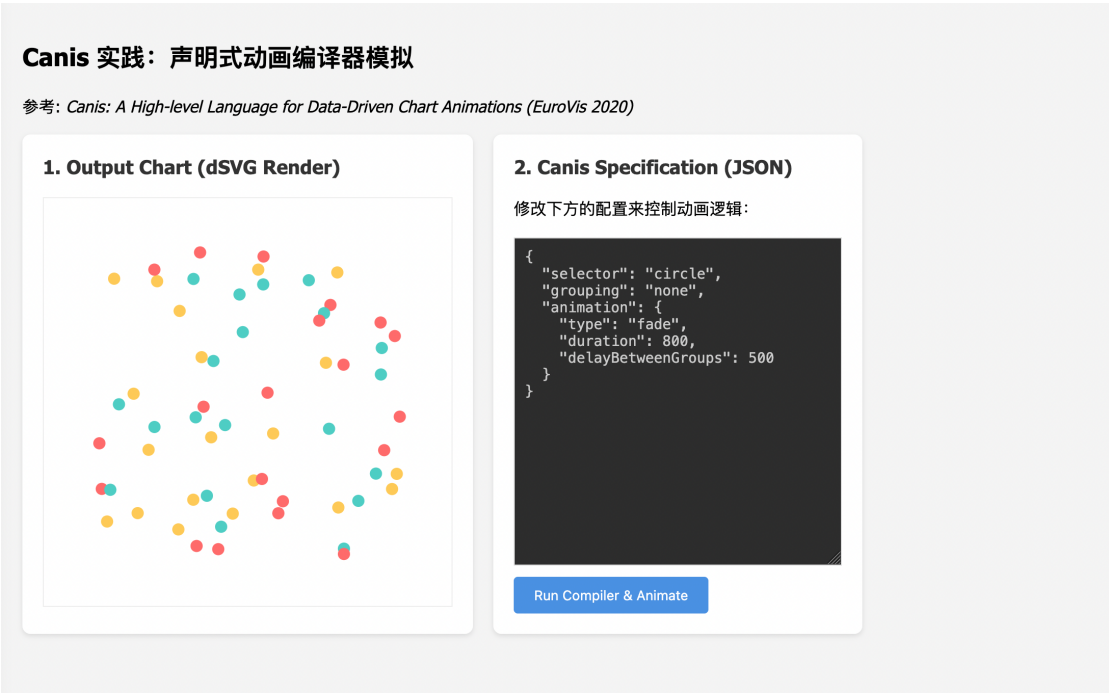

图 2: 修改后

在修改前，颜色 A 的圆点会先逐渐出现，随后是颜色 B 和 C 的圆点; 在修改后，所有的圆点会同时出现，且没有分组动画效果。

4 实验总结与收获

本实验通过模拟 Canis 的编译器机制，展示了如何利用声明式 JSON 配置来驱动数据动画的生成。与传统的命令式编程方式相比，声明式语法大大简化了动画逻辑的编写，使得用户只需关注动画的逻辑而不必担心底层的 DOM 操作。实验中，成功实现了基于数据属性的分组和延迟控制，验证了 Canis 中“划分”与“评估”机制的有效性。