山东大学计算机科学与技术学院

可视化技术课程实验报告

学号：202300130205	姓名：李尚远	班级：23 级数据班
实验题目：可视化相互交织的树木		
实验学时：2	实验日期：2025.11.3	
实验目标：制作一个可视化的相互交织的树的设计 ，且实现力导向布局拖拽等功能。		

作品描述（实验背景、数据集来源、描述思路（为什么用此种可视化形式？能达到什么样的效果？优点？））：

实验背景：

1、数据部分

```
    const treesData = [
        {
            id: "亚洲",
            root: "亚洲",
            nodes: [
                { id: "亚洲", parent: null },
                { id: "东亚", parent: "亚洲" },
                { id: "西亚", parent: "亚洲" },
                { id: "中亚", parent: "亚洲" },
                { id: "中南半岛", parent: "东亚" },
                { id: "中国大陆", parent: "东亚" },
                { id: "朝鲜半岛", parent: "东亚" },
                { id: "俄罗斯", parent: "亚洲" }
            ]
        },
        {
            id: "欧洲",
            root: "欧洲",
            nodes: [
                { id: "欧洲", parent: null },
                { id: "西欧", parent: "欧洲" },
                { id: "中欧", parent: "欧洲" },
                { id: "东欧", parent: "欧洲" },
                {id:"俄罗斯",parent:"东欧"},
                {id:"伊比利亚半岛",parent:"西欧"}
            ]
        },
        {
            id: "美洲",
```

```
            root: "美洲",
            nodes: [
                { id: "美洲", parent: null },
                { id: "北美洲", parent:"美洲" },
                { id: "南美洲", parent: "美洲" },
                { id: "美国", parent: "北美洲" },
                { id: "墨西哥", parent: "北美洲" },
                { id: "巴西", parent: "南美洲" },
                { id: "智利", parent: "南美洲" }
            ]
        }
    ];
```

二、数据处理，构建映射，计算距离等

```
 const treeHierarchy = {}; // 存储每棵树的层级关系：treeHierarchy[树 ID][节点 ID] = 
父节点 ID
        const rootNodes = {}; // 存储每棵树的根节点：rootNodes[树 ID] = 根节点 ID
        treesData.forEach(tree => {
            rootNodes[tree.id] = tree.root;
            const hierarchy = {};
            tree.nodes.forEach(node => {
                hierarchy[node.id] = node.parent;
            });
            treeHierarchy[tree.id] = hierarchy;
        });

        // 3. 计算节点到各树根节点的距离（层级数）
        function getDistanceToRoot(nodeId, treeId) {
            let distance = 0;
            let current = nodeId;
            const hierarchy = treeHierarchy[treeId];
            // 从节点向上遍历到根节点，统计层级数
            while (current !== rootNodes[treeId] && current !== null) {
                current = hierarchy[current];
                distance++;
                // 防止循环引用导致死循环（正常数据不会出现）
                if (distance > 100) break;
            }
            return distance;
        }

        // 4. 处理节点：提取所有不重复的节点，并计算距离最近的根节点
        const allNodes = [...new Set(treesData.flatMap(tree => tree.nodes.map(n => 
n.id)))];
        const nodes = allNodes.map(id => {
            const treeIds = treesData
```

```
            .filter(tree => tree.nodes.some(n => n.id === id))
            .map(tree => tree.id);
        const isRoot = treesData.some(tree => tree.root === id);

        // 关键逻辑：计算到各树根节点的距离，找到最近的根节点
        let minDistance = Infinity;
        let closestRootTreeId = treeIds[0]; // 默认取第一个树 ID
        treeIds.forEach(treeId => {
            const distance = getDistanceToRoot(id, treeId);
            if (distance < minDistance) {
                minDistance = distance;
                closestRootTreeId = treeId;
            }
        });

        return {
            id,
            treeIds,
            isRoot,
            closestRootTreeId // 新增：距离最近的根节点所属树 ID
        };
    });
```

三、创建力导向布局

```
    const simulation = d3.forceSimulation(nodes)
        .force("link", d3.forceLink(links).id(d => d.id).distance(120))
        .force("charge", d3.forceManyBody().strength(-300))
        .force("center", d3.forceCenter(width / 2 - 50, height / 2 - 50))
        .force("collide", d3.forceCollide().radius(60));

// 14. 力导向布局更新
    simulation.on("tick", () => {
        link
            .attr("x1", d => d.source.x)
            .attr("y1", d => d.source.y)
            .attr("x2", d => d.target.x)
            .attr("y2", d => d.target.y);

        node
            .attr("cx", d => d.x)
            .attr("cy", d => d.y);

        label
            .attr("x", d => d.x)
            .attr("y", d => d.y);
    });
```

```
    // 15. 拖拽相关函数
    function dragstarted(event, d) {
        if (!event.active) simulation.alphaTarget(0.3).restart();
        d.fx = d.x;
        d.fy = d.y;
    }

    function dragged(event, d) {
        d.fx = event.x;
        d.fy = event.y;
    }

    function dragended(event, d) {
        if (!event.active) simulation.alphaTarget(0);
        d.fx = null;
        d.fy = null;
    }
```

结果图片：

地区从属关系树可视化图片