D3.js 스터디

4주차 - Force Simulating

이지호

D3-Forces

Forces

- 노드들의 위치를 변경하거나 속도를 변경할 수 있는 function
- 다양한 종류의 forces를 제공
  - Centering : 특정 (x, y)로 끌어당기는 힘
  - Collision : 노드들을 반지름을 가진 원으로 간주해, 이들이 겹치지 않도록 하는 힘
  - Links : 노드 사이를 원하는 거리만큼 유지하려고 하는 힘
  - Many-Body : 모든 노드 사이에서 작용하는 힘(ex : 중력 / 전기적 반발력)
  - Positioning : 특정한 위치에 머무르려고 하는 힘

D3-Forces

Simulation

- d3.forceSimulation : 새로운 simulation object를 만들어 리턴
- simulation.nodes : simulation의 노드를 세팅.
  - 각각의 노드에는 index, x, y, vx, vy, fx, fy 등의 속성이 있음
  - Tick이 끝나면, fx와 fy가 설정된 경우, x와 y가 fx와 fy로 설정되고, vx와 vy가 0이 됨
- simulation.alpha : simulation이 활성화된 정도

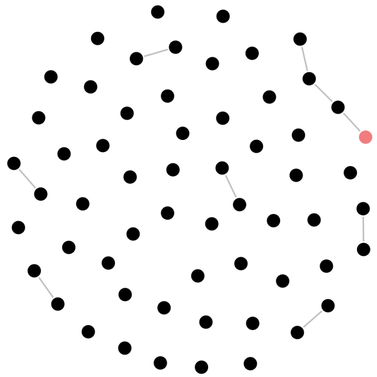
D3-Drag
Drag-And-Drop

- Drag Event : drag event listener에 전달되는 이벤트
  - target : 이벤트가 진행되는 객체
  - x, y : 마우스의 x, y좌표
  - dx, dy : 마우스가 이전 드래그 이벤트 이후로 움직인 x, y 변화량
  - active : 이 target을 제외하고 현재 drag가 진행되는 target의 개수

Temporal Force-Directed Graph

https://observablehq.com/@d3/temporal-force-directed-graph

13:24:00.000Z



Temporal Force-Directed Graph

데이터(graph.json)


```
{
  "nodes":[
    {
      "id":"1467",
      "start":"2009-06-04T09:01:40.000Z",
      "end":"2009-06-04T09:11:20.000Z"
    },
    {
      "id":"1591",
      "start":"2009-06-04T09:01:40.000Z",
      "end":"2009-06-04T09:10:20.000Z"
    },
    {
      "id":"1513",
      "start":"2009-06-04T09:02:20.000Z",
      "end":"2009-06-04T09:02:40.000Z"
    },
    {
      "id":"1568",
      "start":"2009-06-04T09:09:00.000Z",
      "end":"2009-06-04T09:10:20.000Z"
    },
    {
      "id":"1562",
      "start":"2009-06-04T09:11:00.000Z",
      "end":"2009-06-04T09:13:40.000Z"
    },
    {
      "id":"1524",
      "start":"2009-06-04T09:12:40.000Z",
      "end":"2009-06-04T09:13:40.000Z"
    },
    {
      "id":"1771",
      "start":"2009-06-04T09:14:00.000Z",
      "end":"2009-06-04T09:14:20.000Z"
    },
    {
      "id":"1428",
      "start":"2009-06-04T09:14:00.000Z",
      "end":"2009-06-04T09:14:20.000Z"
    },
    {
      "id":"1600",
      "start":"2009-06-04T09:16:20.000Z",
      "end":"2009-06-04T09:17:40.000Z"
    },
    {
      "id":"1523",
      "start":"2009-06-04T09:16:20.000Z",
      "end":"2009-06-04T10:48:40.000Z"
    },
```

Temporal Force-Directed Graph

소스코드(1)

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <script src="https://d3js.org/d3.v6.min.js"></script>
    <title>Graph 예제</title>
</head>
<body>
    <label id="time"></label>
    <button id="button">Pause</button>
    <div class="slidecontainer">
      <input type="range" min="0" max="100" value="0" class="slider" id="myRange">
    </div>
    <svg></svg>
```

Temporal Force-Directed Graph

소스코드(2)

```
<script>
    const width = 1000;
    const height = 600;
    const contains = ({start, end}, time) => {
        return start <= time && time < end;
    }
    const drag = simulation => {
        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;
        }

        return d3.drag()
            .on("start", dragstarted)
            .on("drag", dragged)
            .on("end", dragended);
    }
```

Temporal Force-Directed Graph

소스코드(3)

```
d3.json("graph.json")
.then(org_data => {
    const data = {
        nodes: org_data.nodes.map(({start, end, ...el}) => ({start: new Date(start), end: new Date(end), ...el})),
        links: org_data.links.map(({start, end, ...el}) => ({start: new Date(start), end: new Date(end), ...el})),
    };

    console.log(data);

    const times = d3.scaleTime()
        .domain([d3.min(data.nodes, d => d.start), d3.max(data.nodes, d => d.end)])
        .ticks(1000)
        .filter(time => data.nodes.some(d => contains(d, time)));

    const simulation = d3.forceSimulation()
        .force("charge", d3.forceManyBody())
        .force("link", d3.forceLink().id(d => d.id))
        .force("x", d3.forceX())
        .force("y", d3.forceY())
        .on("tick", ticked);

    const svg = d3.select("svg")
        .attr("viewBox", [-width / 2, -height / 2, width, height]);

    let link = svg.append("g")
        .attr("stroke", "#999")
        .attr("stroke-opacity", 0.6)
        .selectAll("line");

    let node = svg.append("g")
        .attr("stroke", "#fff")
        .attr("stroke-opacity", 1.5)
        .selectAll("circle");

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

        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);
    }
```

Temporal Force-Directed Graph

소스코드(4)

```
function update({nodes, links}) {

    // Make a shallow copy to protect against mutation, while
    // recycling old nodes to preserve position and velocity.
    const old = new Map(node.data().map(d => [d.id, d]));
    nodes = nodes.map(d => Object.assign(old.get(d.id) || {}, d));
    links = links.map(d => Object.assign({}, d));

    node = node
        .data(nodes, d => d.id)
        .join(enter =>
            enter.append("circle")
                .attr("r", 5)
                .call(drag(simulation))
                .call(node => node.append("title").text(d => d.id))
        )
        .on("mouseover", function () {
            d3.select(this).attr("fill", "lightcoral");
        });

    link = link
        .data(links, d => [d.source, d.target])
        .join("line")
        .on("mouseover", function () {
            d3.select(this).attr("fill", "lightcoral");
        });

    simulation.nodes(nodes);
    simulation.force("link").links(links);
    simulation.alpha(1).restart().tick();
    ticked(); // render now!
};

const updatechart = (time) => {
    const nodes = data.nodes.filter(d => contains(d, time));
    const links = data.links.filter(d => contains(d, time));

    update({nodes, links});
};
```

Temporal Force-Directed Graph

소스코드(5)

```
        var timer = setInterval(timeflow, 100);

        range.max = times.length;
        range.addEventListener('input', function(e) {
            idx = e.target.value;

            time = times[idx++];
            timelabel.innerHTML = time.toISOString().split('T')[1];
            updatechart(time);

            clearInterval(timer);
        });

        button.addEventListener('click', function(e) {
            if(e.target.innerHTML === 'Play') {
                e.target.innerHTML = 'Pause';
                timer = setInterval(timeflow, 100);
            } else {
                e.target.innerHTML = 'Play';
                clearInterval(timer);
            }
        })

    });
```