

7.2 JavaScript Code

Working directory:

```
root+---matchPoints*.js
      |---generateRoutine*.js
      |---distance.js
      |---stringFunctionGenerator*.js
```

matchPoints2.js:

```
/*@Filename: matchPoints2.js*/

"use strict";

const fs = require("fs");
const read = fs.readFileSync;
const write = fs.writeFileSync;
const join = require("path").join;
const { eucDis, eucDisXY, eucDisXZ, eucDisZ, eucDisX } =
require("./distance");const objToArray = require("./objToArray");
const getJSON = require("./getJSON");

let start, end;

(function main () {
  try {
    start = [];
    end = [];

    let jsonStart = getJSON("./ferris-wheel-2.json");//any[4]
    let jsonEnd = require("./match-points-2.js");//一维数组
number[3]

    start = objToArray(jsonStart);

    end/*Set*/ = objToArray(jsonEnd)
      .map((arr, i) => [...arr, i])
      .sort((a, b) => eucDisXY(b, [76, 76, 150]) - eucDisXY(a, [76,
76, 150]));

    let matchedStartPoints = [];
    let matchedEndPoints = [];
```

```
let minDistMatches = {};  
let totalDistance = 0;  
let dists = [];  
let variance = 0; //方差  
  
//let clusterSortByHeight = [5, 3, 2, 1, 4];  
  
//for(let clusterIndex = 0; clusterIndex < 5; clusterIndex++) {  
  //let previousCluster = clusterSortByHeight[clusterIndex];  
  for (let endPointIndex = 0; endPointIndex < end.length;  
endPointIndex++) { //遍历终点集合  
    let endPoint = end[endPointIndex];  
    let actualEndPointIndex = endPoint[endPoint.length - 1];  
    //if(cluster[endPointIndex] != previousCluster) { // 对  
应的Cluster  
      // console.log("Not current cluster:",  
cluster[endPointIndex], "!=" , previousCluster);  
      // continue;  
      //}  
      let previousMatch = [];  
      let minDist = Infinity;  
      for (let startPointIndex in start) {  
        let startPoint = start[startPointIndex];  
        let dist = 1 - getCosTheta(startPoint, endPoint);  
        //let dist = eucDisXY(startPoint, endPoint);  
        if (dist < minDist &&  
matchedStartPoints.indexOf(startPoint) === -1 &&  
matchedEndPoints.indexOf(endPoint) === -1) {  
          minDist = dist;  
          previousMatch = [startPoint, endPoint];  
          minDistMatches[startPointIndex] =  
[actualEndPointIndex.toString(), -1];  
        }  
      }  
      if(minDist === Infinity) {  
        continue;  
      }  
      matchedStartPoints.push(previousMatch[0]);  
      matchedEndPoints.push(previousMatch[1]);  
      //console.log(previousMatch);
```

```
        totalDistance += minDist;
        dists.push(minDist);
    }
    //}
    let averageDist = totalDistance / end.length;
    console.log("Total distance:", totalDistance);
    console.log("Average distance:", averageDist);
    let sumOfDeviation = 0;
    dists.forEach(v => {
        //console.log(v, averageDist);
        sumOfDeviation += Math.pow(v - averageDist, 2);
    });
    variance = sumOfDeviation / end.length;
    console.log("Variance:", variance);
    write(join(__dirname, "./match-points-mv-2.json"),
JSON.stringify(minDistMatches));
    } catch (e) {
        console.log(e.message, e.stack);
    }
})();

process.on("uncaughtException", e => {
    console.log(e.message, e.stack);
});

function getCosTheta(st, ed) {
    let [x1, y1, z1] = st;
    let [x2, y2, z2] = ed;
    let A0 = [0, x1 - 76, y1 - 76, z1 - 150]; //hack for reduce
    let B0 = [0, x2 - 76, y2 - 76, z2 - 150]; //hack for reduce
    //console.log(A0[0] * B0[0] + A0[1] * B0[1] + A0[2] * B0[2]);
    return round(
        (A0[1] * B0[1] + A0[2] * B0[2] + A0[3] * B0[3])
        /
        (
            Math.sqrt(A0.reduce((sum, num) => sum + Math.pow(num, 2)))
            *
            Math.sqrt(B0.reduce((sum, num) => sum + Math.pow(num, 2)))
        )
    );
};
```

```
}

function round(num) {
  let integer, decimal;
  if (num >= 0) {
    integer = Math.floor(num);
    decimal = num - integer;
  } else {
    integer = Math.ceil(num);
    decimal = num - integer;
  }
  decimal = Math.round(decimal * 1e12) / 1e12;
  return integer + decimal;
}
```

generateRoutine2.js:

```
/*@Filename: generateRoutine2.js*/

"use strict";

const fs = require("fs");
const read = fs.readFileSync;
const write = fs.writeFileSync;
const join = require("path").join;

const stringFuncGen = require("./stringFunctionGenerator2.js");

let riseFuncStrs = [];

const objToArray = require("./objToArray");
const getJSON = require("./getJSON");

let presetSleepTime = [6.5, 4, 1, 8.5, 0];
let presetPaddingTime = [6, 8, 10, 4, 12];

function main () {
  let colors = objToArray(require("./point-color.json"));
  let start = [];
  const matches = getJSON("./match-points-mv-2.json");
```

```

    let jsonStart = getJSON("./ferris-wheel-2.json"),
        end = require("./dragon-points.js");
    start = objToArray(jsonStart);
    for(let startPointIndex in matches) {
        let startPoint = start[startPointIndex];
        let endPoint = end[matches[startPointIndex][0]];
        let sleepTime = 2;//presetSleepTime[matches[startPointIndex][1]
- 1];
        let paddingTime =
0;//presetPaddingTime[matches[startPointIndex][1] - 1];
        let cluster = -1; //matches[startPointIndex][1];
        let color = colors[startPointIndex];
        //console.log(startPoint, endPoint, color, sleepTime,
matches[startPointIndex][0], paddingTime);
        //console.log(matches[startPointIndex][0]);
        //console.log(startPointIndex);
        riseFuncStrs[startPointIndex] = stringFuncGen(startPoint,
endPoint, color, sleepTime, startPointIndex, paddingTime, cluster);
    }
    write(join(__dirname, "./move-funcs.json"),
JSON.stringify(riseFuncStrs));
    write(join(__dirname, "./move-funcs.js"), "let mvFxStrs = " +
JSON.stringify(riseFuncStrs) + ";");
}

main();

```

distance.js:

```

/*@Filename: distance.js*/

"use strict";

function eucDis(point1, point2) {
    //console.log(point1, point2);
    let x1 = point1[0], y1 = point1[1], z1 = point1[2];
    let x2 = point2[0], y2 = point2[1], z2 = point2[2];
    return Math.sqrt(
        Math.pow(x1 - x2, 2) + Math.pow(y1 - y2, 2) + Math.pow(z1 - z2,
2)
    );
}

```

```
    );  
}  
  
function eucDisXY(point1, point2) {  
    //console.log(point1, point2);  
    let x1 = point1[0], y1 = point1[1];  
    let x2 = point2[0], y2 = point2[1];  
    return Math.sqrt(  
        Math.pow(x1 - x2, 2) + Math.pow(y1 - y2, 2)  
    );  
}  
  
function eucDisXZ(point1, point2) {  
    //console.log(point1, point2);  
    let x1 = point1[0], z1 = point1[2];  
    let x2 = point2[0], z2 = point2[2];  
    return Math.sqrt(  
        Math.pow(x1 - x2, 2) + Math.pow(z1 - z2, 2)  
    );  
}  
  
function eucDisZ(point1, point2) {  
    //console.log(point1, point2);  
    let z1 = point1[2];  
    let z2 = point2[2];  
    return Math.sqrt(  
        Math.pow(z1 - z2, 2)  
    );  
}  
  
function eucDisX(point1, point2) {  
    //console.log(point1, point2);  
    let x1 = point1[0];  
    let x2 = point2[0];  
    return Math.sqrt(  
        Math.pow(x1 - x2, 2)  
    );  
}  
  
module.exports = {
```

```
    eucDis,  
    eucDisXY,  
    eucDisXZ,  
    eucDisZ,  
    eucDisX  
}
```

stringFunctionGenerator2.js:

```
/*@Filename: stringFunctionGenerator2.js*/  
  
"use strict";  
  
const list = [107, 458, 397, 150, 474, 231, 386, 201, 204, 119, 126, 425,  
236, 294, 461, 68, 268]; // 1s delay  
const list2 = [238, 80, 381, 9, 231, 206, 219, 269, 304, 302, 454]; //  
1.5s delay  
const list3 = [103, 414, 404, 261, 424, 112, 374, 270, 325, 448, 473,  
299, 337, 196, 447, 379, 198]; // 1s in advance  
const list4 = [195, 154, 339, 280, 175, 288, 255, 240, 132]; // 1.5s in  
advance  
  
const _slope = [334, 324, 345, 344, 292, 290, 195, 194, 235, 233, 253,  
56, 297, 291, 42, 18, 275, 274, 228, 59, 237, 236, 271, 255, 455, 441,  
413, 174, ...("194 384 370 414 113 176 121 103 62 184 169 404 348 121  
12 449 445 347".split(" ").map(Number))]; //set slope to  $\pm 2$   
  
const _slope2 = [42, 22, 45, 35, 374, 372, 60, 33, 131, 118, 259, 56,  
397, 185, 411, 199, 45, 44, 111, 64, 303, 302, 246, 245, 277, 247]; //set  
slope to  $\pm 1.7$   
  
const zeroSlope = [80]; //set slope to 0  
  
let k = 0,  
    sleepTimes = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1,  
1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9],  
    slopes = [-1.6, -1.5, -1.3, -1, -0.8, -0.6, -0.3, 0.3, 0.6, 0.8, 1,  
1.3, 1.5, 1.6],  
    velocities = [2.8, 4];
```

```
function stringFuncGen (startPoint, endPoint, color, sleepTime,
endIndex, lifting, cluster, latency) {
  endIndex = Number(endIndex);
  if(list.indexOf(Number(endIndex)) !== -1) {
    sleepTime += 1;
    lifting -= 1;
  }
  if(list2.indexOf(Number(endIndex)) !== -1) {
    sleepTime += 1.5;
    lifting -= 1.5;
  }
  if(list3.indexOf(Number(endIndex)) !== -1) {
    sleepTime -= 1;
    lifting += 1;
  }
  if(list4.indexOf(Number(endIndex)) !== -1) {
    sleepTime -= 1.5;
    lifting += 1;
  }
  sleepTime += sleepTimes[k % sleepTimes.length];
  var slope = slopes[k % slopes.length];
  var velocity = velocities[k % velocities.length];
  k++;
  if(typeof color === "undefined")
    color = "#fff";
  if(typeof latency !== "number")
    latency = 8;
  var _l = latency;
  latency -= lifting;
  let duration = latency - sleepTime;
  if(latency !== duration) {
    //throw Error("f**k");
  }
  if(_slope.indexOf(endIndex) !== -1) {
    slope = ((_slope.indexOf(endIndex)) % 2 - .5) * 2 * 2;
    console.log(endIndex, slope);
  } else if(_slope2.indexOf(endIndex) !== -1) {
    slope = ((_slope2.indexOf(endIndex)) % 2 - .5) * 2 * 1.7;
    console.log(endIndex, slope);
  }
}
```



```

    if(zeroSlope.indexOf(endIndex) !== -1) {
        slope = 0;
        console.log(endIndex, slope);
    }
    let x1 = startPoint[0], y1 = startPoint[1], z1 = startPoint[2];
    let x2 = endPoint[0], y2 = endPoint[1], z2 = endPoint[2];
    if(y1 > 52) {
        slope = -Math.abs(slope);
    } else if(y1 < 52) {
        slope = Math.abs(slope);
    }
    //z2 += 24;
    let useBezier = false; // && cluster === 2 && k % 2 === 1; //
Middle-Layer
    return `    var tx = arguments[0];
    var _tx = tx;
    tx -= ${ sleepTime };
    if(tx > ${ duration }) {
        return [${ x2 }, ${ y2 }, ${ z2 }, "${ color }"];
    }
    if(tx <= 0) {
        return [${ x1 }, ${ y1 }, ${ z1 }, "${ color }"];
    }

    ${ useBezier ? "" : "///" }var bPoint = bezier(tx, ${ duration });

    ${ useBezier ? "" : "///" }var bx = bPoint.x,
    ${ useBezier ? "" : "///" }    bz = bPoint.y;

    var deltaY = ${ velocity } * ${ slope } * (Math.pow((tx - ${ duration
/ 2 } ), 2) - Math.pow(${ duration / 2 }, 2));

    //console.log("${ endIndex }");

    return [${
        (x2 - x1) / (useBezier ? 1 : duration)
    } * ${ useBezier ? "bx" : "tx" } + ${ x1 }, ${
        (y2 - y1) / duration
    } * tx + ${ y1 } + deltaY, ${
        (z2 - z1) / (useBezier ? 1 : duration)

```

```
    } * ${ useBezier ? "bz" : "tx" } + ${ z1 }, "${  
        color  
    }"];`;  
}  
  
module.exports = stringFuncGen;
```

7.3 Coordinates