راهنمای توسعه

فایل index.js

این فایل نقطه ی شروع برنامه است و ارتباط برنامه را با رابط کاربری برقرار میکند.

وظایف :

- افزودن event listener ها
- افزودن و تبدیل تصاویر ورودی به canvas
- نمایش رنگ های خروجی الگوریتم
- مدیریت رویداد ها

فایل chart.js

این فایل پیکسل ها را به صورت سه بعدی نمایش می دهد

وظایف :

- افزون پیکس ها به فضای سه بعدی
- تغییر رنگ پیکسل ها با تغییر خوشه

فایل kmeans.js

این فایل وظیفه ی اجرای الگوریتم های خوشه یابی را بر عهده دارد.

وظایف :

- تبدیل پیکسل های تصویر به نقاط سه بعدی
- انتخاب خوشه ها به صورت رندوم
- انتخاب خوشه برای هر پیکسل
- به روز رسانی مکان خوشه

تابع imageDataToVertexes :

این تابع در دو حلقه ی تو در تو تمام پیکسل های عکس را به RGB تبدیل کرده و یک راس برای آن ایجاد می کند.

```
function imageDataToVertexes(imageData) {
    vertexes = [];

    for (let x = 0; x < imageData.width; x++) {
        for (let y = 0; y < imageData.height; y++) {
            let index = (y * imageData.width + x) * 4;
            let red = imageData.data[index];
            let green = imageData.data[index + 1];
            let blue = imageData.data[index + 2];
            // let alpha = imageData.data[index + 3];

            vertexes.push({
                pos: new THREE.Vector3(red - 127, green - 127, blue - 127),
                color: 'rgb(' + red + ',' + green + ',' + blue + ')'
            });
        }
    }
}
```

تابع randomCluster :

این تابع به تعداد خوشه ها یک راس را انتخاب و با موقعیت ان یک راس خوشه ایجاد میکند سپس خوشه ها را به صورت لیبل به صفحه اضافه میکند.

```
function randomCluster() {
    clusters = [];

    for (let i = 0; i < clusterCount; i++) {
        let index = Math.floor(Math.random() * (vertexes.length - 1));

        let x = vertexes[index].pos.x;
        let y = vertexes[index].pos.y;
        let z = vertexes[index].pos.z;

        let color = `rgb(${x + 127},${y + 127},${z + 127})`;

        clusters.push({
            pos: new THREE.Vector3(x, y, z),
            sum: new THREE.Vector3(0, 0, 0),
            dataCount: 0,
            label: $('<div></div>').addClass('color').css('background-color', color),
            color: color
        });
    }

    appendLabels(clusters.map(c => c.label));
}
```

تابع choseClusters :

این تابع برای همه راس ها نزدیک ترین خوشه را پیدا می کند و اگر خوشه ی راسی تغییر کند در اخرین تغییر ثبت می شود.

```
function choseClusters() {
    for (let i = 0; i < vertexes.length; i++) {
        let minDistance = clusters[0].pos.distanceTo(vertexes[i].pos);
        let minDistanceIndex = 0;

        for (let j = 1; j < clusterCount; j++) {
            let distance = clusters[j].pos.distanceTo(vertexes[i].pos);
            if (distance < minDistance) {
                minDistance = distance;
                minDistanceIndex = j;
            }
        }

        if (vertexes[i].cluster != minDistanceIndex)
            lastChange = step_Num;
        vertexes[i].cluster = minDistanceIndex;
    }
}
```

تابع updateCentroid :

این تابع ابتدا مختصات هر راس را در کلاستور آن جمع میکند سپس برای هر کلاستور حاصل جمع را به تعداد خوشه ها تقسیم میکند و در نهایت مقدار به دست امده را به عنوان مختصات جدید خوشه ثبت می کند.

```
function updateCentroid() {
    // sum of each cluster points
    let i;
    for (i = 0; i < vertexes.length; i++) {
        let index = vertexes[i].cluster;
        clusters[index].sum = clusters[index].sum.add(vertexes[i].pos);
        clusters[index].dataCount += 1;
    }
    for (i = 0; i < clusterCount; i++) {
        if (clusters[i].dataCount)
            clusters[i].pos = clusters[i].sum.divideScalar(clusters[i].dataCount);
        clusters[i].sum = new THREE.Vector3(0, 0, 0);
        clusters[i].dataCount = 0;
        // update color whit pos
        let x = Math.floor(clusters[i].pos.x) + 127;
        let y = Math.floor(clusters[i].pos.y) + 127;
        let z = Math.floor(clusters[i].pos.z) + 127;
        let rgb = `rgb(${x},${y},${z})`;
        clusters[i].color = rgb;
        clusters[i].label.css('background-color', rgb);
    }
}
```

تابع runKmeans :

این تابع تا زمانی که خوشه ی راسی دچار تغییر شود خوشه ی تمام راس را تایین میکند و سپس مرکز ثقل هر خوشه را به روز می کند.

```
function runKmeans(draw) {
    let loop = setInterval(() => {
        if (step_Num - lastChange < 2) {
            step_Num++;

            choseClusters();
            updateCentroid();
            draw();
        }
        else {
            step_Num = 0;
            lastChange = 0;
            clearInterval(loop);
        }
    }, 100);
}
```