计算机图形学课程

实验报告

学号： **19114801**

班级： 计科二班

姓名： 刘爱兵

目　录

实验一：直线段扫描转换算法

1.实验目的与要求：

利用 **DDALine**、**Bresenham** 算法实现线段的扫描转换，通过 **JavaScript** 实现其算法，画出线段，分析课本上算法的局限性。

2.实验内容和实验步骤：

（1）测试老师的代码
（2）测试书本代码
（3）理解算法核心思想
（4）完善书本代码的不足之处
（5）开始书写代码，测试、调试，实现结果

3.实验结果：

（1）老师代码结果

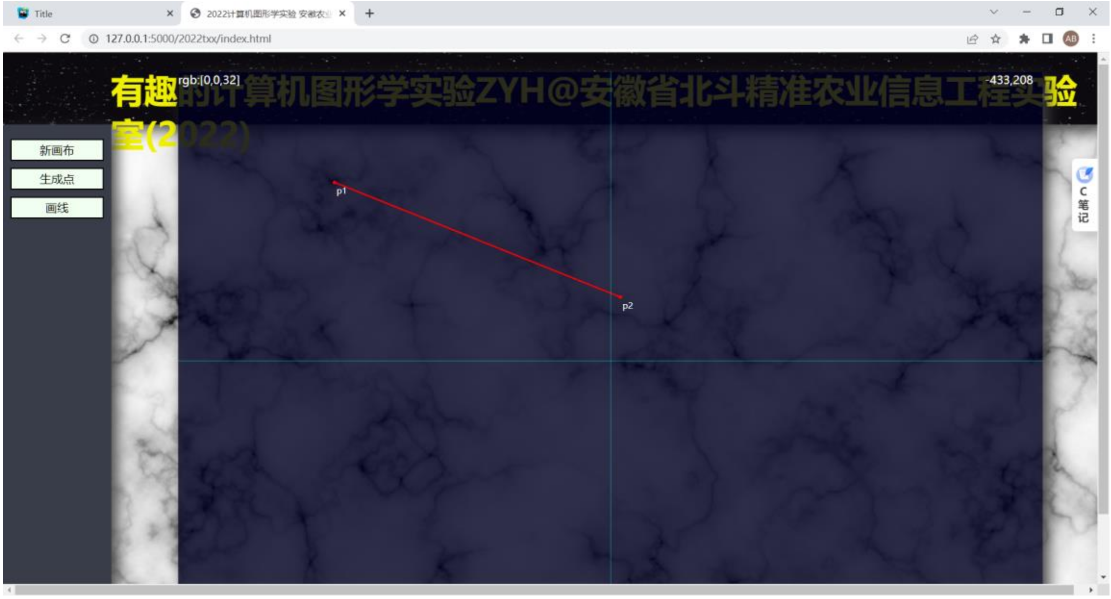

Title
2022计算机图形学实验 安徽农业
127.0.0.1:5000/2022txx/index.html
有趣的计算机图形学实验ZYH@安徽省北斗精准农业信息工程实验室(2022)
rgb:[0,0,32]
-433,208
新画布
生成点
画线
p1
p2
C
笔记

图 1 **DDALine** 结果

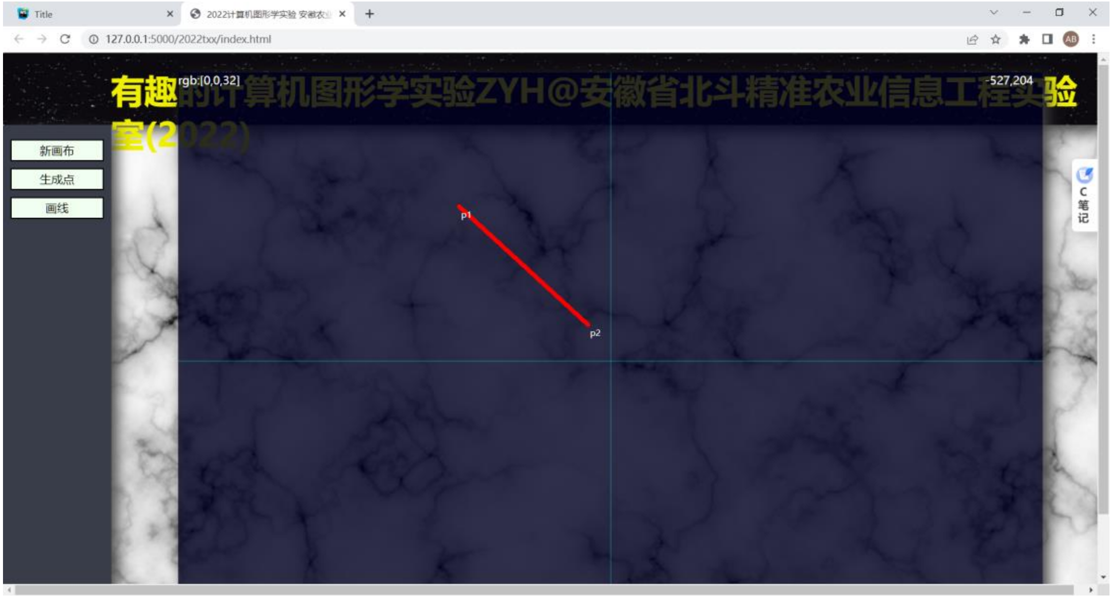

Title
2022计算机图形学实验 安徽农
127.0.0.1:5000/2022txx/index.html
有趣的计算机图形学实验ZYH@安徽省北斗精准农业信息工程实验室(2022)
rgb:[0,0,32]
527,204
新画布
生成点
画线
p1
p2

图 2　**Bresenham 结果**

（2）书本代码结果

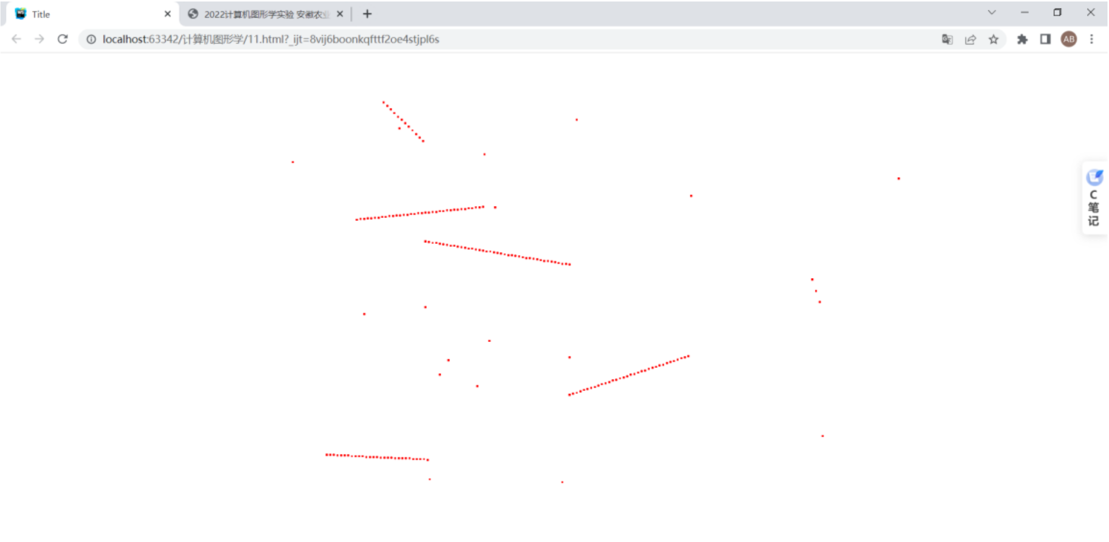

Title
2022计算机图形学实验 安徽农
localhost:63342/计算机图形学/11.html?_ijt=8vij6boonkqfttf2oe4stjpl6s

图 3 **书本代码 DDALine 结果**

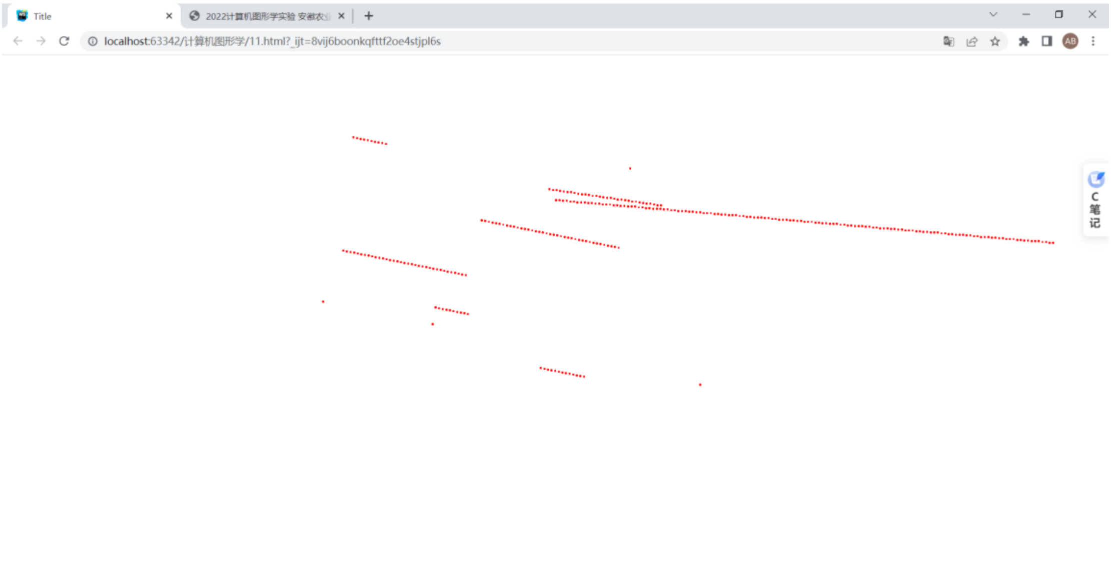

Title
2022计算机图形学实验 安徽农
localhost:63342/计算机图形学/11.html?_ijt=8vij6boonkqfttf2oe4stjpl6s
C
笔
记

图 4　**书本代码 Bresenham 结果**

（3）自己设计代码

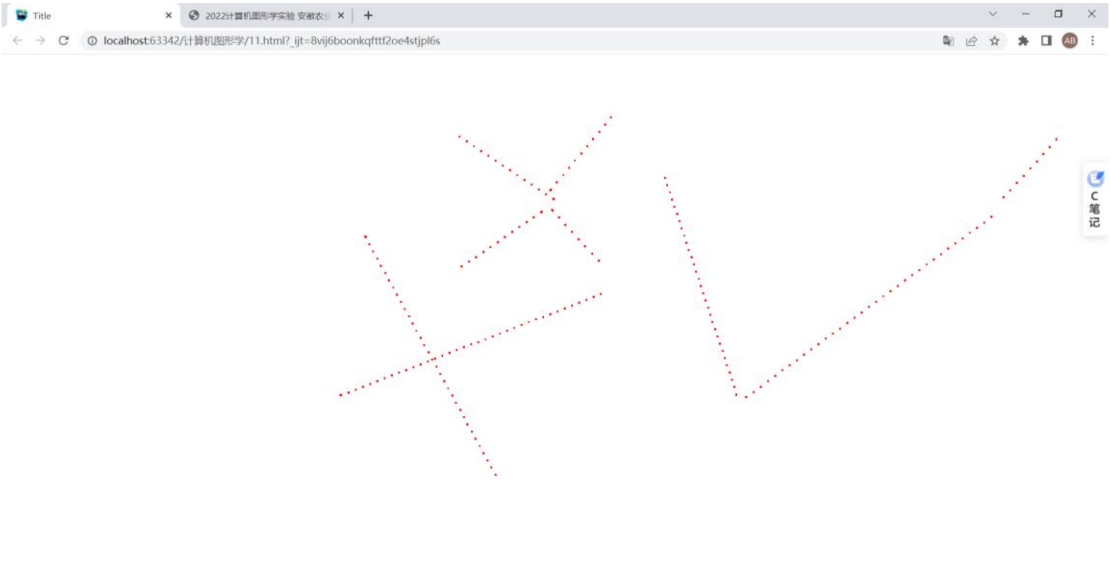

Title
2022计算机图形学实验 安徽农
localhost:63342/计算机图形学/11.html?_ijt=8vij6boonkqfttf2oe4stjpl6s
C
笔
记

图 5 **自己代码 DDALine 结果**

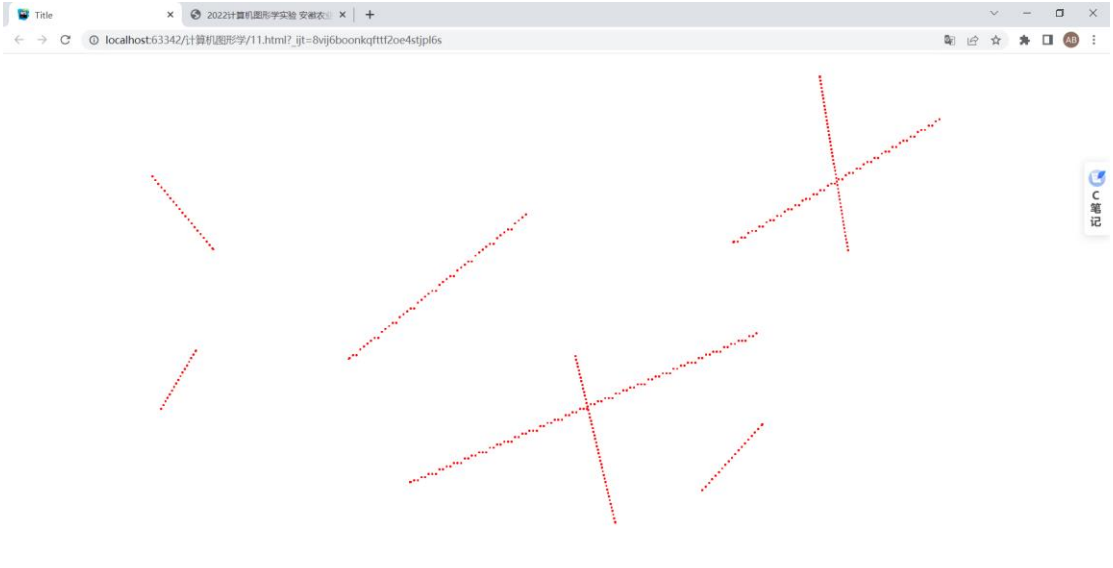

Title
2022计算机图形学实验 安徽农
localhost:63342/计算机图形学/11.html?_ijt=8vij6boonkqfttf2oe4stjpl6s
C
笔
记

图 6 **自己代码 Bresenham 结果**

4.心得体会：

通过分析老师的代码，可以发现老师的算法还是有很多可以学习的地方、界面和模块设计为后期统一管理提供了很大的便捷和扩展性。

课本上 DDALine 算法的局限性：

依旧存在大量的浮点运算，每步都需四舍五入取整。

课本上只实现了第二个点在第一个点右侧才可以画出，，我们需要实现所有方向的问题

课本上 Bresenham 算法的局限性：

理论上是连续的，但实际是离散的，同样存在浮点运算

课本只是介绍了一个象限的情况，我们么需要考虑四个象限的问题

5.源程序：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script>
    var f;
    // 按下鼠标
    window.onmousedown = function (e) {
        if (undefined === f) {
            f = {
                x: e.clientX,
                y: e.clientY
            }
            createPointer(f.x, f.y);
            return;
        }
        // 第二次点击触发，绘制第二个点
        createDDALine(f.x, f.y, e.clientX, e.clientY);
        f = undefined;
    }

    // 在指定位置绘制一个点
    function createPointer(x, y) {
        var html = '<div class="pointer" style="width:3px; height:3px;
position:absolute; ' +
            'background:red; border-radius:50%; top:' + y + 'px; left:' +
x + 'px;" ></div>';
        document.body.innerHTML += html;
    }

    //-----------------------DDA 算法
--------------------------------------------//
    // 两点之间画线,老师的算法
    function DDALine(x0, y0, x1, y1) {
        // createPointer(x0, y0); //创建第一个点
        // createPointer(x1, y1); //创建第二个点
        var dx = x1 - x0;
        var dy = y1 - y0;
        var k, x, y, xs, ys, xe, ye, xx, yy;
        if (Math.abs(dx) > Math.abs(dy)) {
            k = dy / dx;
            if (x1 > x0) {
                xs = x0;
                xe = x1;
                y = y0;
                for (x = xs; x <= xe; x++) {
                    yy = parseInt(y + 0.5);
                    createPointer(x, yy);
                    y = y + k;
                }
            }
            if (x0 > x1) {
                xs = x1;
                xe = x0;
                y = y1;
                for (x = xs; x <= xe; x++) {
                    yy = parseInt(y + 0.5);
                    createPointer(x, yy);
                    y = y + k;
                }
            }
        }

        if (Math.abs(dy) > Math.abs(dx)) {
            k = dx / dy;
            if (y1 > y0) {
                ys = y0;
                ye = y1;
                x = x0;
                for (y = ys; y <= ye; y++) {
                    xx = parseInt(x + 0.5);
                    createPointer(xx, y);
                    x = x + k;
                }
            }
            if (y0 > y1) {
                ys = y1;
                ye = y0;
                x = x1;
                for (y = ys; y <= ye; y++) {
                    xx = parseInt(x + 0.5);
                    createPointer(xx, y);
                    x = x + k;
                }
            }
        }
    }

    // 两点之间画线
    function createDDALine(x0, y0, x1, y1) {
        // createPointer(x0, y0); //创建第一个点
        // createPointer(x1, y1); //创建第二个点
        // 计算出倾斜角
        var dx, dy;
        var rX, rY;
```

```
        var i;
        if (x0 < x1) { // b点在a点的右边
            dx = x1 - x0;
            rX = 1;
        } else { // b点在a点的左边
            dx = x0 - x1;
            rX = -1;
        }
        if (y0 < y1) { // b点在a点的下面
            dy = y1 - y0;
            rY = 1;
        } else {
            dy = y0 - y1;
            rY = -1;
        }
        var k = dy / dx; // 角度比
        // 绘制直线
        var maxX = Math.abs(x0 - x1);
        var maxY = Math.abs(y0 - y1);
        if (maxX > maxY) {
            for (i = 1; i < maxX; i += 10) {
                var tempY = i * k; //在x轴上进行移动画点，没有进行精度运算（取0.5整的）

                var tempYchange = parseInt(tempY * rY + 0.5);
                createPointer(x0 + i * rX, y0 + tempYchange);
            }
        } else {
            for (i = 1; i < maxY; i += 10) {
                var tempX = i / k; //在y轴上进行移动画点
                var tempXchange = parseInt(tempX * rX + 0.5);//数值微分改进

                createPointer(x0 + tempXchange, y0 + i * rY);
            }
        }
    }

    function book_DDALine(x0, y0, x1, y1) {
        createPointer(x0, y0); //创建第一个点
        createPointer(x1, y1); //创建第二个点
        var dx, dy, y, k;
        dx = x1 - x0;
        dy = y1 - y0;
        k = dy / dx;
        y = y0;
        for (x = x0; x <= x1; x = x + 5) {
            createPointer(x, parseInt(y + 0.5));
            y = y + k;
        }
    }

    //-----------------------Bresenham 算法
--------------------------------------------//
    function createBresenham(x0, y0, x1, y1) {
        var dx, dy;
        var rX, rY;  //方向
        var i;
        if (x0 < x1) { // b 点在 a 点的右边
            dx = x1 - x0;
            rX = 1;
        } else { // b 点在 a 点的左边
            dx = x0 - x1;
            rX = -1;
        }
        if (y0 < y1) { // b 点在 a 点的下面
            dy = y1 - y0;
            rY = 1;
        } else {
            dy = y0 - y1;
            rY = -1;
        }
        var k = dy / dx; // 角度比
        // 绘制直线
        var e = -0.5;
        var xchange = x0;
        var ychange = y0;
        var maxX = Math.abs(x0 - x1);
        var maxY = Math.abs(y0 - y1);
        if (maxX > maxY) {   //x 轴为步进
            for (i = 1; i < maxX; i += 5) {
                var tempY = i * k;
                createPointer(xchange, ychange);
                e = e + k;
                xchange = x0 + i * rX;
                if (e >= 0) {
                    ychange = y0 + tempY * rY;
                    e = e - 1;
                }
            }
        } else {       //y 轴为步进
            for (i = 1; i < maxY; i += 5) {
                var tempX = i / k;
                createPointer(xchange, ychange);
                e = e + k;
                ychange = y0 + i * rY;
                if (e >= 0) {
                    xchange = x0 + tempX * rX;
                    e = e - 1;
                }
            }
        }
    }

    function book_Bresenham(x0, y0, x1, y1) {
        var x, y, dx, dy;
        var k, e;
        dx = x1 - x0;
        dy = y1 - y0;
        k = dy / dx;
        e = -0.5;
        x = x0;
        y = y0;
        for (i = 0; i <= dx; i = i + 5) {
            createPointer(x, y);
            x = x + 5;
            e = e + k;
            if (e >= 0) {
                y++;
                e = e - 1;
            }
        }
    }
</script>
</body>
</html>
```

实验二：区域填充算法

1.实验目的与要求：

利用种子递归的区域填充算法实现多边形填充，通过 JavaScript 实现其算法。

2 实验内容和实验步骤：

（1）学习种子递归区域填充填充算法
（2）参考书上的递归算法
（3）实现算法结果

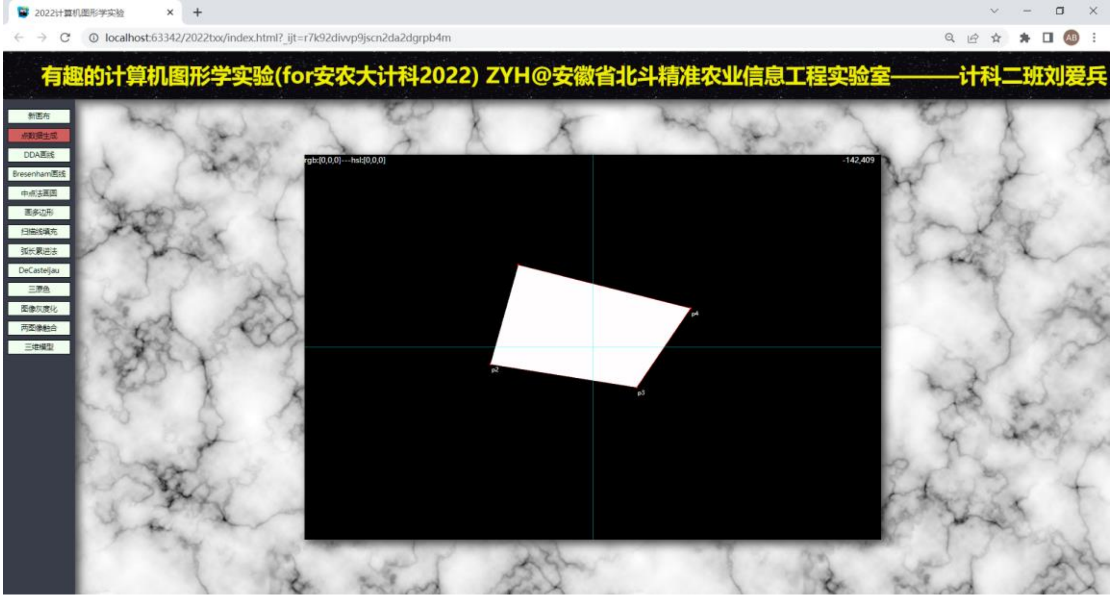
1. 实验结果：

（1）参考书上的代码

```
void BoundaryFill4(int x, iny, int boundarycolor, int newcolor) {
               int color = getpixel(x, y);
               if (color != newcolor && color != boundarycolor) {
                   drawpixel(x, y, newcolor);
                   BoundryFill4(x, y + 1, boundarycolor, newcolor);
                   BoundryFill4(x, y - 1, boundarycolor, newcolor);
                   BoundryFill4(x - 1, y, boundarycolor, newcolor);
                   BoundryFill4(x + 1, y, boundarycolor, newcolor);
               }
           }
```

说明:递归算法进行取像素点颜色比较，满足取点颜色与边框颜色和填充颜色不一致时进行颜色填充 ，移动像素点，重复此过程

（2）本设计采用以扫描固定形状的图片边框进行种子填充算法结果



2022计算机图形学实验
localhost:63342/2022txx/index.html?_ijt=r7k92divvp9jscn2da2dgrpb4m
有趣的计算机图形学实验(for安农大计科2022) ZYH@安徽省北斗精准农业信息工程实验室———计科二班刘爱兵
新画布
点数据生成
DDA画线
Bresenham画线
中点法画圆
画多边形
扫描线填充
弧长累进法
DeCasteljau
三原色
图像灰度化
两图像融合
三维模型
rgb:[0,0,0]---hsl:[0,0,0]
-142,409
p2
p3
p4

图 1 **多边形绘制与填充**

4.心得体会：

实现区域填充，首先要绘制图形边框，实现画图功能，借鉴于此，我们采用绘制图片的边框，给定边框的颜色进行填充来模拟绘制图形，实验在设计的过程中学会了种子区域填充算法。

5.源程序：

```
//扫描线填充算法
function scanPolyFillproc() {
    if (pt.length < 3) {
        alert("需要先生成 3 个以上不共线的点数据")
        return
    }
    var ymin, ymax;
    var rct = calcRect(pt);
    var lines = [];
    ymin = rct[1];
    ymax = rct[3];

    var cnt = pt.length;
    for (var i = 0; i < cnt - 1; i++) {
        lines.push([pt[i][0], pt[i][1], pt[i + 1][0], pt[i + 1][1]]);
    }
    lines.push([pt[cnt - 1][0], pt[cnt - 1][1], pt[0][0], pt[0][1]]);

    var xroot = [],
        xr;
    lncnt = lines.length;
    for (var y = ymin; y < ymax; y++) {
        for (var i = 0; i < lncnt; i++) {
            if (judgeCross(lines[i], y)) {
                xr = getXRoot(lines[i], y);
                xroot.push(Math.round(xr));
            }
        }
        xroot.sort(function(a, b) {
            return a - b
        }); //数值数组通过比值函数升序排列，改为 return b-a 则为降序
        if (xroot.length >= 3 && xroot.length % 2 == 1) {
            xroot = distinct(xroot);
        }
        var segcnt = int(xroot.length / 2);
        for (var i = 0; i < segcnt; i++) {
            BresenhamLine(xroot[i * segcnt], y, xroot[i * segcnt + 1], y, "#80000080");
        }
        xroot.length = 0;
    }
}
//数组去重
function distinct(arr) {
    return Array.from(new Set(arr))
}


// drawpoly
function drawPoly(poly, color, size) {
    pcnt = poly.length;
    p0 = poly[0];
    for (var i = 1; i < pcnt; i++) {
        p1 = poly[i];
        BresenhamLine(p0[0], p0[1], p1[0], p1[1], color, size);
        p0 = p1;
    }
    p1 = poly[0];
    BresenhamLine(p0[0], p0[1], p1[0], p1[1], color, size);
}

function calcRect(poly) {
    cnt = poly.length;
    xmin = 10000, xmax = -10000, ymin = 10000, ymax = -10000;
    for (i = 0; i < cnt; i++) {
        x = poly[i][0];
        y = poly[i][1];
        if (x < xmin) {
            xmin = x;
        }
        if (y < ymin) {
            ymin = y;
        }
        if (x >= xmax) {
            xmax = x;
        }
        if (y >= ymax) {
            ymax = y;
        }
    }
    //console.log(cnt);
    return [xmin, ymin, xmax, ymax];
}

function drawRect(x0, y0, x1, y1, color) {
    var poly = [
        [x0, y0],
        [x0, y1],
        [x1, y1],
        [x1, y0]
    ];
    drawPoly(poly, color);
}

//ln=[{x： x0,y:y0},{x： x0,y:y0}]
function getXRoot(ln, y) {
    x0 = ln[0], y0 = ln[1];
    x1 = ln[2], y1 = ln[3];
    if (x1 == x0) {
        return x0;
    }
    k = (y1 - y0) / (x1 - x0);
    b = y0 - k * x0;
    x = (y - b) / k;
    return x;
}

function judgeCross(myln, lny) {
    p0y = myln[1];
    p1y = myln[3];
    if ((p0y - lny) * (p1y - lny) <= 0) {
        return true;
    }
    return false;
}
```

实验三：弧长累进法判断点与多边形关系

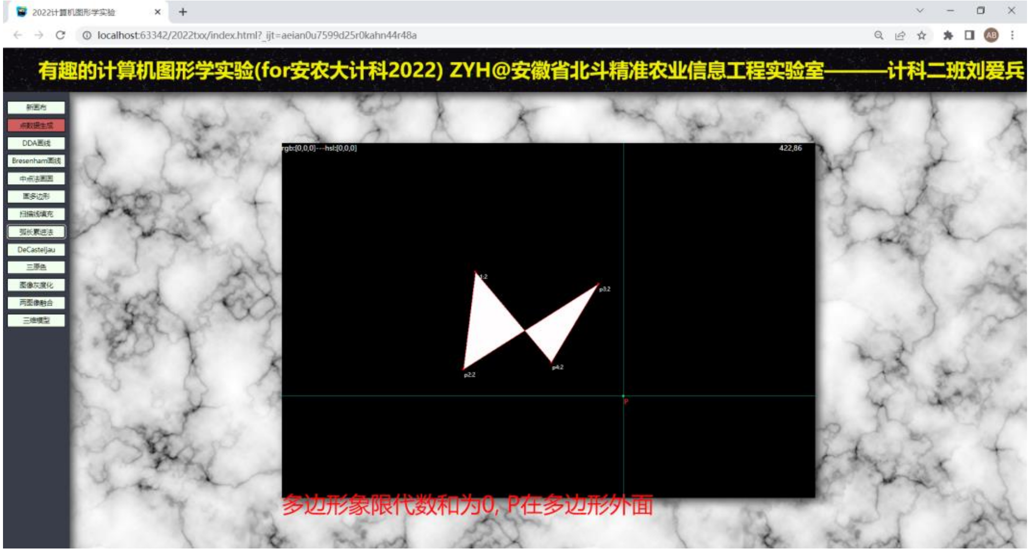
1.实验目的与要求：

检测点与多边形关系，利用弧长累进法快速判断给定点与给定多边形的关系，通过 JavaScript 实现其算法。

2.实验内容和实验步骤：

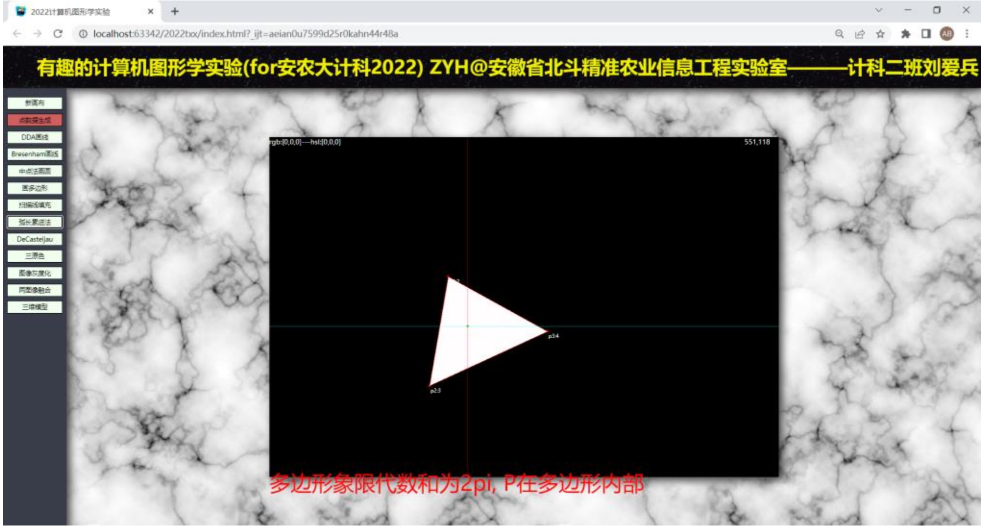
(1)绘制一个多边形
(2)绘制一个点，判断点与多边形的关系

3.实验结果：


2022计算机图形学实验
localhost:63342/2022txx/index.html?_ijt=aeian0u7599d25r0kahn44r48a
有趣的计算机图形学实验(for安农大计科2022) ZYH@安徽省北斗精准农业信息工程实验室———计科二班刘爱兵
新画布
点数据生成
DDA画线
Bresenham画线
中点法画圆
画多边形
扫描线填充
弧长累进法
DeCasteljau
三原色
图像灰度化
两图像融合
三维模型
rgb:[0,0,0]---hsl:[0,0,0]
422,86
1:2
p2:2
p3:2
p4:2
P

多边形象限代数和为0, P在多边形外面

图 1 **p** 点在外判断


2022计算机图形学实验
localhost:63342/2022txx/index.html?_ijt=aeian0u7599d25r0kahn44r48a
有趣的计算机图形学实验(for安农大计科2022) ZYH@安徽省北斗精准农业信息工程实验室———计科二班刘爱兵
新画布
点数据生成
DDA画线
Bresenham画线
中点法画圆
画多边形
扫描线填充
弧长累进法
DeCasteljau
三原色
图像灰度化
两图像融合
三维模型
rgb:[0,0,0]---hsl:[0,0,0]
551,118
p2:3
p3:4

多边形象限代数和为2pi, P在多边形内部

图 2 **p 点在内判断**

4.心得体会：

判断点在多边形的内部还是外边，以被测点 O 为坐标原点，将平面划分为 4 个象限，对每个多边形顶点 P[i]，计算其所在的象限，然后顺序访问多边形的各个顶点 P[i]，分析 P[i]和 P[i+1]。

5.源程序：

```
function huchangleiji() {
        var checkpt;
        listleng = pt.length;
        if (listleng >= 4) {
            checkpt = pt[listleng - 1];
            pt.length = pt.length - 1;
            var poly = [];
            for (var i = 0; i < pt.length; i++) {
                poly.push([pt[i][0], pt[i][1]]);
            }

            drawPoly(poly, "#ff0000");
            xoySetPixel(checkpt[0], checkpt[1], "#00ff00", 5);
            nX = checkpt[0];
            nY = checkpt[1];
            var divbiaozhu = select("#biaozhup" + listleng);
            divbiaozhu.html("P");
            pt.push(pt[0]);
            divAxisX.position(0, oY - nY);
            divAxisY.position(oX + nX, 0);
            var dx = nX,
                dy = nY;
            for (var i = 0; i < pt.length - 1; i++) {
                pt[i][0] = pt[i][0] - dx;
                pt[i][1] = pt[i][1] - dy;
            }
            oX = oX + nX;
            oY = oY - nY;
            panduandianweizhi(pt);

        }
    }

    function panduandianweizhi(poly) {
        var xoy = [];
        var listleng = poly.length;
        for (var i = 0; i < listleng - 1; i++) {
            if (poly[i][0] >= 0 && poly[i][1] >= 0) xoy.push(1);
            if (poly[i][0] < 0 && poly[i][1] >= 0) xoy.push(2);
            if (poly[i][0] < 0 && poly[i][1] < 0) xoy.push(3);
            if (poly[i][0] >= 0 && poly[i][1] < 0) xoy.push(4);
            var divbz = select("#biaozhup" + (i + 1));
            divbz.html("p" + (i + 1) + ":" + xoy[i]);
        }
        var sum = 0;
        var PI = 3.14,
            PI2 = 1.57;
        xoy.push(xoy[0]);
        for (var i = 0; i < xoy.length - 1; i++) {
            if (xoy[i] == 1 && xoy[i + 1] == 1) sum += 0;
            if (xoy[i] == 1 && xoy[i + 1] == 2) sum += PI2;
            if (xoy[i] == 1 && xoy[i + 1] == 3) sum += -PI;
            if (xoy[i] == 1 && xoy[i + 1] == 4) sum += -PI2;
            if (xoy[i] == 2 && xoy[i + 1] == 1) sum += -PI2;
            if (xoy[i] == 2 && xoy[i + 1] == 2) sum += 0;
            if (xoy[i] == 2 && xoy[i + 1] == 3) sum += PI2;
            if (xoy[i] == 2 && xoy[i + 1] == 4) sum += -PI;
            if (xoy[i] == 3 && xoy[i + 1] == 1) sum += -PI;
            if (xoy[i] == 3 && xoy[i + 1] == 2) sum += -PI2;
            if (xoy[i] == 3 && xoy[i + 1] == 3) sum += 0;
            if (xoy[i] == 3 && xoy[i + 1] == 4) sum += PI2;
            if (xoy[i] == 4 && xoy[i + 1] == 1) sum += PI2;
            if (xoy[i] == 4 && xoy[i + 1] == 2) sum += -PI;
            if (xoy[i] == 4 && xoy[i + 1] == 3) sum += -PI2;
            if (xoy[i] == 4 && xoy[i + 1] == 4) sum += 0;
        }
        var res;
        if (Math.abs(sum) < 0.1) res = "在多边形外面";
        if (Math.abs(sum) < 6.3 && Math.abs(sum) > 6) res = "在多边形里面
";
        divFootHint.html("多边形象限代数和为" + sum + ", P" + res);
    }
```

实验四：二阶 Bezier 曲线的实现

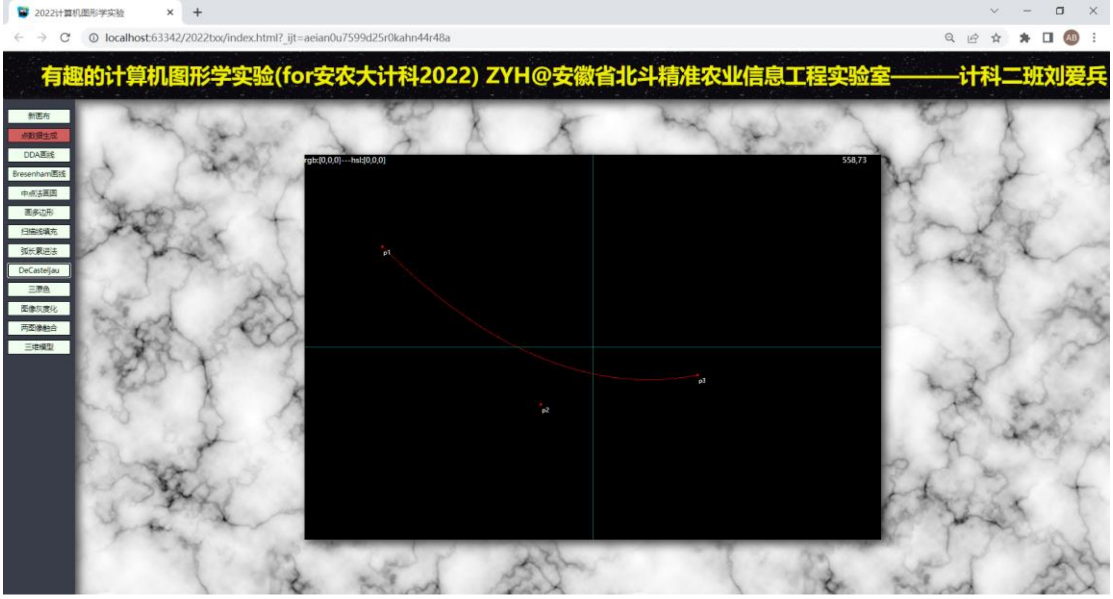
1.实验目的与要求：

掌握基于 Bezier 曲线的定义和 De Casteljau 方法的 Bezier 曲线生成原理，至少会用一种方法，通过 JavaScript 画出二次 Bezier 曲线。

2.实验内容和实验步骤：

（1）学习 DeCasteljau 方法的 Bezier 曲线生成原理
（2）书写算法绘制曲线

3.实验结果：



2022计算机图形学实验

localhost:63342/2022txx/index.html?_ijt=aeian0u7599d25r0kahn44r48a

有趣的计算机图形学实验(for安农大计科2022) ZYH@安徽省北斗精准农业信息工程实验室———计科二班刘爱兵

rgb:[0,0,0]---hsl:[0,0,0]

558,73

图 1 **DeCasteljau 方法**

4.心得体会：

学习 De Casteljau 算法，重在理解使用递归推理每一阶的情况，然后理解算法的核心思想，最后我们使用 js 进行递归处理，学习了很多，也了解了很多。

5.源程序：

```
t = 0;
function deCasteljauproc() {
    n = pt.length;
    for (t = 0; t <= 1; t = t + 0.001) {
        res = deCasteljau(pt, t);
        xoySetPixel(int(res[0][0]), int(res[0][1]), "#ff0000", 1)
    }
}

function deCasteljau(pt, t) {
    curp = [];
    nextp = [];
    curp = [...pt]
    while (1) {
        n = curp.length;
        if (n == 1) break;
        k = 0;
        for (i = 0; i < n - 1; i++) {
            xi = curp[i][0] * (1 - t) + curp[i + 1][0] * t;
            yi = curp[i][1] * (1 - t) + curp[i + 1][1] * t;
            nextp.push([xi, yi])
        }
        if (nextp.length == 1) {
            return nextp;
        }
        if (nextp.length > 1) {
            curp = [...nextp];
            nextp.length = 0;
        }
    }
}
```

实验五：颜色模型

1.实验目的与要求：

了解 **RGB** 颜色模型，通过 **JavaScript** 实现三原色显示、图像灰度化和两图像融合。

2.实验内容和实验步骤：

（1）实现三原色显示
（2）实现图像灰度化
（3）实现两图像融合

3.实验结果：

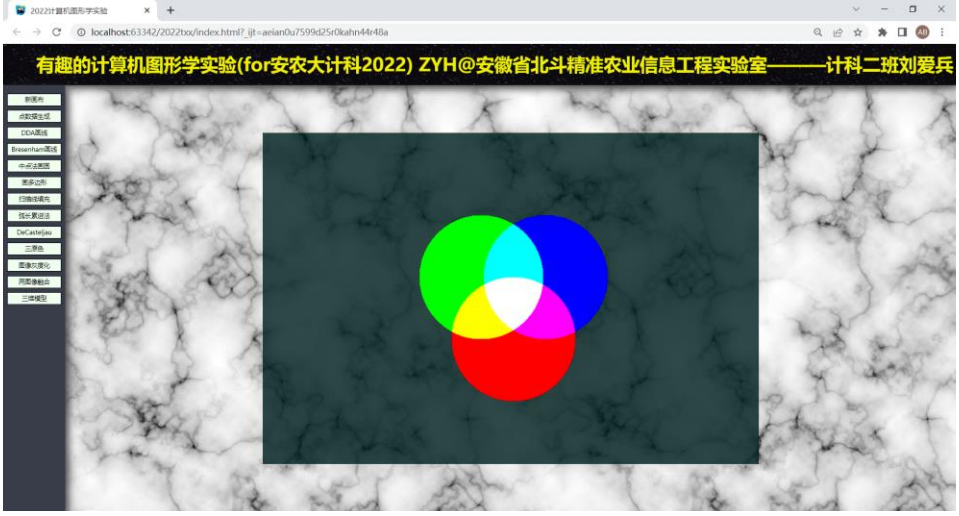

2022计算机图形学实验
localhost:63342/2022txx/index.html?_ijt=aeian0u7599d25r0kahn44r48a
有趣的计算机图形学实验(for安农大计科2022) ZYH@安徽省北斗精准农业信息工程实验室———计科二班刘爱兵
新画布
点数据生成
DDA画线
Bresenham画线
中点法画圆
画多边形
扫描线填充
弧长累进法
DeCasteljau
三原色
图像灰度化
两图像融合
三维模型

图 1 三原色显示

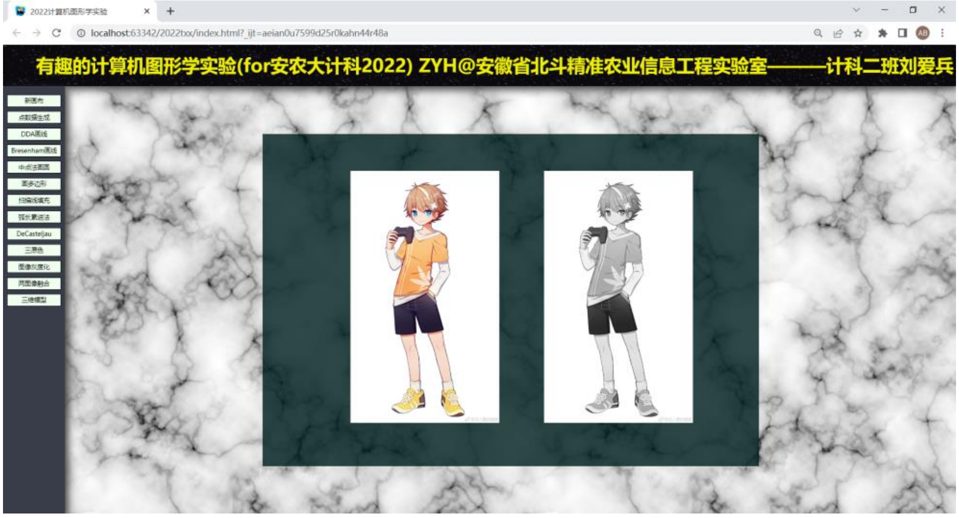

2022计算机图形学实验
localhost:63342/2022txx/index.html?_ijt=aeian0u7599d25r0kahn44r48a
有趣的计算机图形学实验(for安农大计科2022) ZYH@安徽省北斗精准农业信息工程实验室———计科二班刘爱兵
新画布
点数据生成
DDA画线
Bresenham画线
中点法画圆
画多边形
扫描线填充
弧长累进法
DeCasteljau
三原色
图像灰度化
两图像融合
三维模型

图 2 图像灰度化

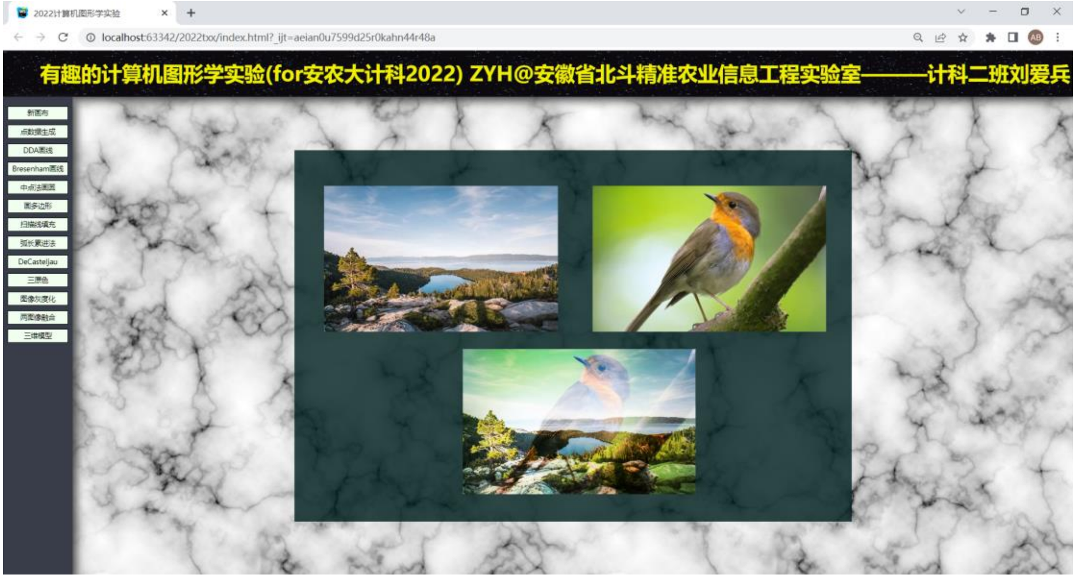

2022计算机图形学实验
localhost:63342/2022txx/index.html?_ijt=aeian0u7599d25r0kahn44r48a
AB
有趣的计算机图形学实验(for安农大计科2022) ZYH@安徽省北斗精准农业信息工程实验室———计科二班刘爱兵
新画布
点数据生成
DDA画线
Bresenham画线
中点法画圆
画多边形
扫描线填充
弧长累进法
DeCasteljau
三原色
图像灰度化
两图像融合
三维模型

图 3 **两图像融合**

4.心得体会：

刚开始不理解三原色，图像灰度化，特别是图像融合，最终在查看老师的之后有很好的理解，理解三原色之后就可以很好的实现其中的效果，灰度化则是去色的过程，最终麻烦的是如何实现图像融合，在理解图层之后，将两种图像淡化处理叠加之后得到新的图像就是图像的融合，在这个过程中也是学会了不少，谢谢老师的参考。

5.源程序：

```
function initmergecir(w, h) {
    mergecir = createImage(w, h);
    mergecir.loadPixels();
    for (j = 0; j < h; j++) {
        for (i = 0; i < w; i++) {
            setImgColor(mergecir, i, j, 0, 0, 0, 255);
        }
    }
    mergecir.updatePixels();
    image(mergecir, 0, 0, mergecir.width, mergecir.height);
}

function drawRedCircle(x0, y0, r) {
    var w = 2 * r,
        clr;
    for (var j = 0; j < w; j++) {
        for (var i = 0; i < w; i++) {
            if (judgeInCircle(r, r, r, i, j)) {
                clr = getImgColor(mergecir, x0 - r + i, y0 - r + j);
                setImgColor(mergecir, x0 - r + i, y0 - r + j, 255 + clr[0],
0 + clr[1], 0 + clr[2], 255);
            }
        }
    }
    mergecir.updatePixels();
    image(mergecir, 100 + 0, 0, mergecir.width, mergecir.height);
}

function drawGreenCircle(x0, y0, r) {
    var w = 2 * r,
        clr;
    for (var j = 0; j < w; j++) {
        for (var i = 0; i < w; i++) {
            if (judgeInCircle(r, r, r, i, j)) {
                clr = getImgColor(mergecir, x0 - r + i, y0 - r + j);
                setImgColor(mergecir, x0 - r + i, y0 - r + j, 0 + clr[0], 255
+ clr[1], 0 + clr[2], 255);
            }
        }
    }
    mergecir.updatePixels();
    image(mergecir, 100 + 0, 0, mergecir.width, mergecir.height);
}


function drawBlueCircle(x0, y0, r) {
    var w = 2 * r,
        clr;
    for (var j = 0; j < w; j++) {
        for (var i = 0; i < w; i++) {
            if (judgeInCircle(r, r, r, i, j)) {
                clr = getImgColor(mergecir, x0 - r + i, y0 - r + j);
                setImgColor(mergecir, x0 - r + i, y0 - r + j, 0 + clr[0], 0
+ clr[1], 255 + clr[2], 255);
            }
        }
    }
    mergecir.updatePixels();
    image(mergecir, 100 + 0, 0, mergecir.width, mergecir.height);
}

function judgeInCircle(x0, y0, r, x, y) {
    if ((x0 - x) * (x0 - x) + (y0 - y) * (y0 - y) <= r * r) return true;
    return false;
}

function clearme() {
    initmergecir(400, 400);
}
function showImage(img, posx, posy) {
    image(img, posx, posy, img.width, img.height);
}

function getImgColor(img, x, y) {
    w = img.width;
    index = (x + y * w) * 4;
    red = img.pixels[index];
    green = img.pixels[index + 1];
    blue = img.pixels[index + 2];
    alpha = img.pixels[index + 3];
    return [red, green, blue, alpha];
}

var a = 0;
var rvsflag = 0;

function mergeIt() {
    var clr1, clr2, r, g, b;
    if (rvsflag == 0) {
        a = a + 0.05;
        if (a >= 1) rvsflag = 1;
    }
    if (rvsflag == 1) {
        a = a - 0.05;
        if (a <= 0) rvsflag = 0;
    }
    var img1 = imgflower;
    var img2 = imgbird;
    img1.loadPixels();
    img2.loadPixels();
    var w1 = img1.width;
    var h1 = img1.height;
    var w2 = img2.width;
    var h2 = img2.height;
    var img3 = createImage(w2, h1);
    img3.loadPixels();
    for (var i = 0; i < w2; i++) {
        for (var j = 0; j < h1; j++) {
            clr1 = getImgColor(img1, i, j);
            clr2 = getImgColor(img2, i, j);
            r = clr1[0] * a + clr2[0] * (1 - a);
            g = clr1[1] * a + clr2[1] * (1 - a);
            b = clr1[2] * a + clr2[2] * (1 - a);
            setImgColor(img3, i, j, r, g, b, 255);
        }
    }
    img3.updatePixels();
    image(img3, oX, oY, img3.width, img3.height);
}

function grayIt() {
    var img = imgbird;
    var clr, gray;
    img.loadPixels();
    w = img.width;
    h = img.height;
    var imggray = createImage(w, h);
    imggray.loadPixels();
    for (var i = 0; i < w; i++) {
        for (var j = 0; j < h; j++) {
            clr = getImgColor(img, i, j);
            gray = clr[0] * 0.3 + clr[1] * 0.6 + clr[2] * 0.1;
            setImgColor(imggray, i, j, gray, gray, gray, 255);
        }
    }
    imggray.updatePixels();
    image(imggray, oX, oY, imggray.width, imggray.height);
}

function setImgColor(img, x, y, red, green, blue, alpha) {
    w = img.width;
    index = (x + y * w) * 4;
    img.pixels[index] = red;
    img.pixels[index + 1] = green;
    img.pixels[index + 2] = blue;
    img.pixels[index + 3] = alpha;
}

function drawRGBBlock(xl, yt, w) {
    var r, g, b;
    var x, y;
    var color;
    for (var x = 0, r = 0; r < 256; r++, x++) {
        for (var y = 0, g = 0; g < 256; g++, y++) {
            color = rgb2color(r, g, 0);
            xoySetPixel(xl + x, yt + y, color);
        }
    }
}
```

实验六：基于 Three.js 三维物体实现（综合实验）

1.实验目的与要求：

在掌握图形学基础的基础上，综合利用点、线、面生成算法和光照模型，生成具有真实感三维图，利用 Three.js 实现其动画效果，掌握 Camera、Render、Scene、对象、光照之间的关系，为计算机三维设计打下基础。

2.实验内容和实验步骤：

（1）引用 three.js
（2）尝试简单的模型生成
（3）实现基础的光照灯效果
（4）选择合适模型进行贴图，实现最终效果

3.实验结果：

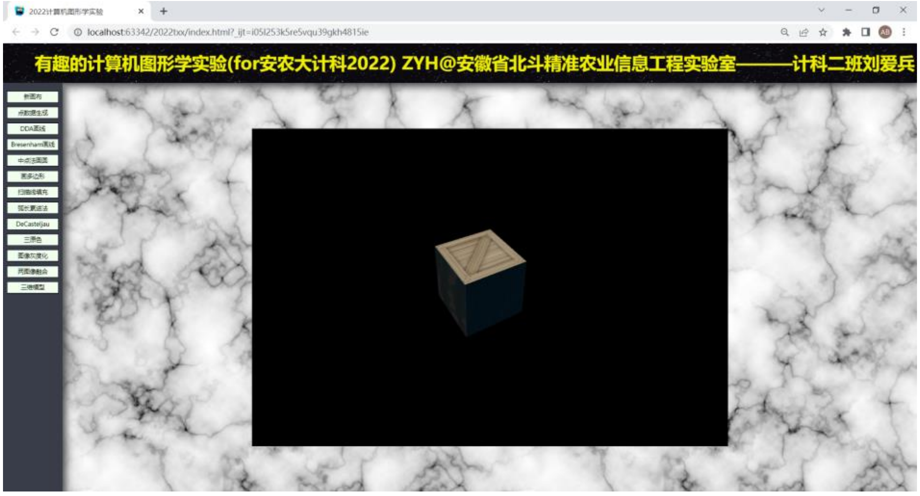

2022计算机图形学实验
localhost:63342/2022txx/index.html?_ijt=i05l253k5re5vqu39gkh4815ie
有趣的计算机图形学实验(for安农大计科2022) ZYH@安徽省北斗精准农业信息工程实验室———计科二班刘爱兵
新画布
点数据生成
DDA画线
Bresenham画线
中点法画圆
画多边形
扫描线填充
弧长累进法
DeCasteljau
三原色
图像灰度化
两图像融合
三维模型

图 1 三维图

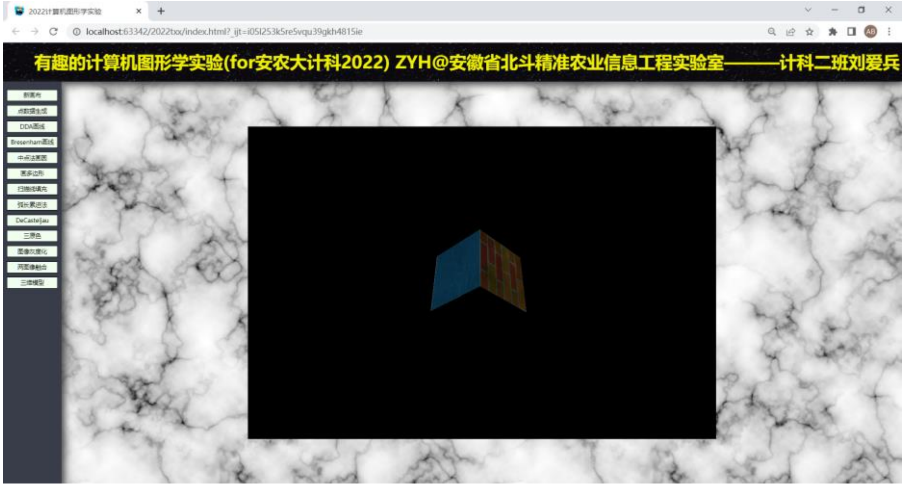

2022计算机图形学实验
localhost:63342/2022txx/index.html?_ijt=i05l253k5re5vqu39gkh4815ie
有趣的计算机图形学实验(for安农大计科2022) ZYH@安徽省北斗精准农业信息工程实验室———计科二班刘爱兵
新画布
点数据生成
DDA画线
Bresenham画线
中点法画圆
画多边形
扫描线填充
弧长累进法
DeCasteljau
三原色
图像灰度化
两图像融合
三维模型

图 2 三维图

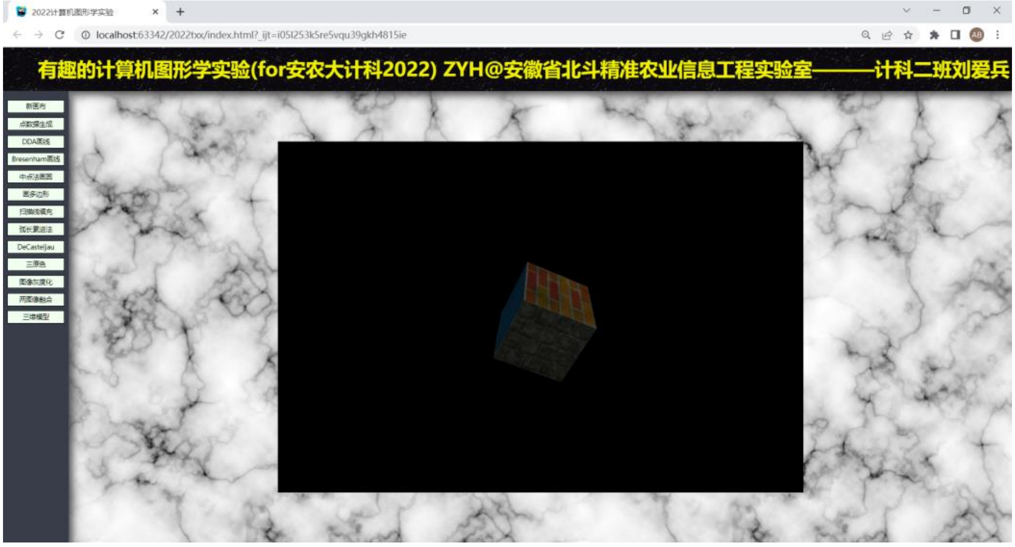

2022计算机图形学实验
localhost:63342/2022txx/index.html?_ijt=i05l253k5re5vqu39gkh4815ie
有趣的计算机图形学实验(for安农大计科2022) ZYH@安徽省北斗精准农业信息工程实验室———计科二班刘爱兵
新画布
点数据生成
DDA画线
Bresenham画线
中点法画圆
画多边形
扫描线填充
弧长累进法
DeCasteljau
三原色
图像灰度化
两图像融合
三维模型

图 3 三维图

4.心得体会：

刚开始应用 three.js 的实例的时候，很多对应的函数都不太熟悉，在反复尝试之后，学会了利用点、线、面生成算法和光照模型，生成具有真实感三维图，实现动态的三维模型的旋转，光照等，之前也非常对 three.js 有兴趣，现在这一次有机会接触到这个 three.js，很感谢了。

5.源程序：

```
<script>
    let ele = document.querySelector('#Wrapper')
    let game=new TextureMapping(ele)
</script>
```

```
let that = null
class TextureMapping {
  constructor(container) {
    that = this
    this.objects = []

    // render
    this.renderer = new THREE.WebGLRenderer();
    this.renderer.setSize( window.innerWidth, window.innerHeight );
    container.appendChild( this.renderer.domElement );

    // scene
    this.scene = new THREE.Scene();
    window.scene = this.scene

    // camera
    this.camera = new THREE.PerspectiveCamera( 70, window.innerWidth /
window.innerHeight, 1, 1000);
    this.scene.add(this.camera);

    // light
    let light = new THREE.DirectionalLight( 0xffffff );
    light.position.set( 0, 1, 1 ).normalize();
    this.scene.add(light);

    // 3D object
    this.objects.push(this.difImgBoxByUV())
    this.scene.add(this.objects[0]);

    // run
    this.run()
  }

  // 纯色的 Box
  pureColorBox() {
    let geometry = new THREE.CubeGeometry( 10, 10, 10);
    let material = new THREE.MeshPhongMaterial( { ambient: 0x050505, color:
0x0033ff, specular: 0x555555, shininess: 30 } );
    let mesh = new THREE.Mesh(geometry, material );
    mesh.position.z = -50;
    mesh.position.x = -10
    return mesh
  }

  // 6 个面相同的 Box
  unityImgBox() {
    let geometry = new THREE.CubeGeometry( 10, 10, 10);
    var material = new THREE.MeshPhongMaterial( { map:
THREE.ImageUtils.loadTexture('./img/crate.jpg') } );
    let mesh = new THREE.Mesh(geometry, material );
    mesh.position.z = -50;
    mesh.position.x = 10
    return mesh
  }

  // 6 个面不相同的 Box
  difImgBox() {
    let geometry = new THREE.CubeGeometry( 10, 10, 10);
    var material1 = new THREE.MeshPhongMaterial( { map:
THREE.ImageUtils.loadTexture('./img/crate.jpg') } );
    var material2 = new THREE.MeshPhongMaterial( { map:
THREE.ImageUtils.loadTexture('./img/bricks.jpg') } );
    var material3 = new THREE.MeshPhongMaterial( { map:
THREE.ImageUtils.loadTexture('./img/clouds.jpg') } );
    var material4 = new THREE.MeshPhongMaterial( { map:
THREE.ImageUtils.loadTexture('./img/stone-wall.jpg') } );
    var material5 = new THREE.MeshPhongMaterial( { map:
THREE.ImageUtils.loadTexture('./img/wood-floor.jpg') } );
    var material6 = new THREE.MeshPhongMaterial( { map:
THREE.ImageUtils.loadTexture('./img/water.jpg') } );
    let materials = [material1, material2, material3, material4, material5,
material6]
    let mesh = new THREE.Mesh(geometry, materials );
    mesh.position.z = -50;
    mesh.position.x = 0
    mesh.position.y = 10
    return mesh
  }

  // 6个面不相同的Box通过uv
  difImgBoxByUV() {
    let geometry = new THREE.CubeGeometry(10, 10, 10);
    // console.error(geometry, 'geometrygeometry')

    // 1) 加载UV贴图
    var material = new THREE.MeshPhongMaterial( { map:
THREE.ImageUtils.loadTexture('./img/texture-atlas.jpg') } );

    // 2) 创建贴图的6个子图
    var bricks = [new THREE.Vector2(0, .666), new THREE.Vector2(.5, .666),
new THREE.Vector2(.5, 1), new THREE.Vector2(0, 1)];
    var clouds = [new THREE.Vector2(.5, .666), new THREE.Vector2(1, .666),
new THREE.Vector2(1, 1), new THREE.Vector2(.5, 1)];
    var crate = [new THREE.Vector2(0, .333), new THREE.Vector2(.5, .333),
new THREE.Vector2(.5, .666), new THREE.Vector2(0, .666)];
    var stone = [new THREE.Vector2(.5, .333), new THREE.Vector2(1, .333),
new THREE.Vector2(1, .666), new THREE.Vector2(.5, .666)];
    var water = [new THREE.Vector2(0, 0), new THREE.Vector2(.5, 0), new
THREE.Vector2(.5, .333), new THREE.Vector2(0, .333)];
    var wood = [new THREE.Vector2(.5, 0), new THREE.Vector2(1, 0), new
THREE.Vector2(1, .333), new THREE.Vector2(.5, .333)];
```

```
    // 3) 清除现有的 UV 映射
    geometry.faceVertexUvs[0] = []

    // 4) 为图元指定纹理
    geometry.faceVertexUvs[0][0] = [ bricks[0], bricks[1], bricks[3] ];
    geometry.faceVertexUvs[0][1] = [ bricks[1], bricks[2], bricks[3] ];
    geometry.faceVertexUvs[0][2] = [ clouds[0], clouds[1], clouds[3] ];
    geometry.faceVertexUvs[0][3] = [ clouds[1], clouds[2], clouds[3] ];
    geometry.faceVertexUvs[0][4] = [ crate[0], crate[1], crate[3] ];
    geometry.faceVertexUvs[0][5] = [ crate[1], crate[2], crate[3] ];
    geometry.faceVertexUvs[0][6] = [ stone[0], stone[1], stone[3] ];
    geometry.faceVertexUvs[0][7] = [ stone[1], stone[2], stone[3] ];
    geometry.faceVertexUvs[0][8] = [ water[0], water[1], water[3] ];
    geometry.faceVertexUvs[0][9] = [ water[1], water[2], water[3] ];
    geometry.faceVertexUvs[0][10] = [ wood[0], wood[1], wood[3] ];
    geometry.faceVertexUvs[0][11] = [ wood[1], wood[2], wood[3] ];

    let mesh = new THREE.Mesh(geometry,  material);
    mesh.position.z = -50;
    mesh.position.x = 0
    mesh.position.y = -10
    return mesh
  }

  run() {
    that.objects.forEach(itemMesh => {
      itemMesh.rotation.x += .02;
      itemMesh.rotation.y += .02;
    });

    that.renderer.render(that.scene, that.camera)
    requestAnimationFrame(that.run)
  }
}
```