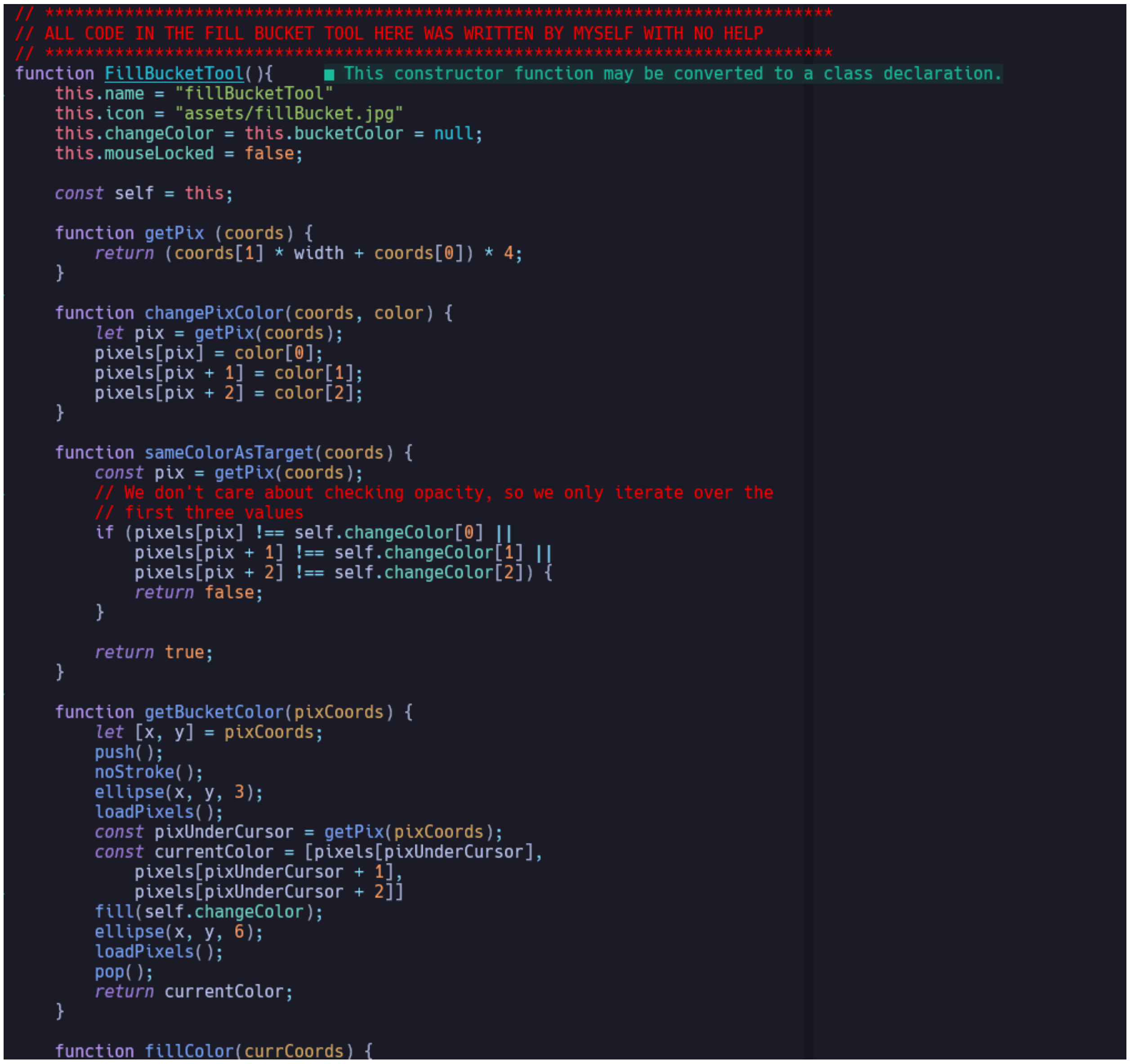
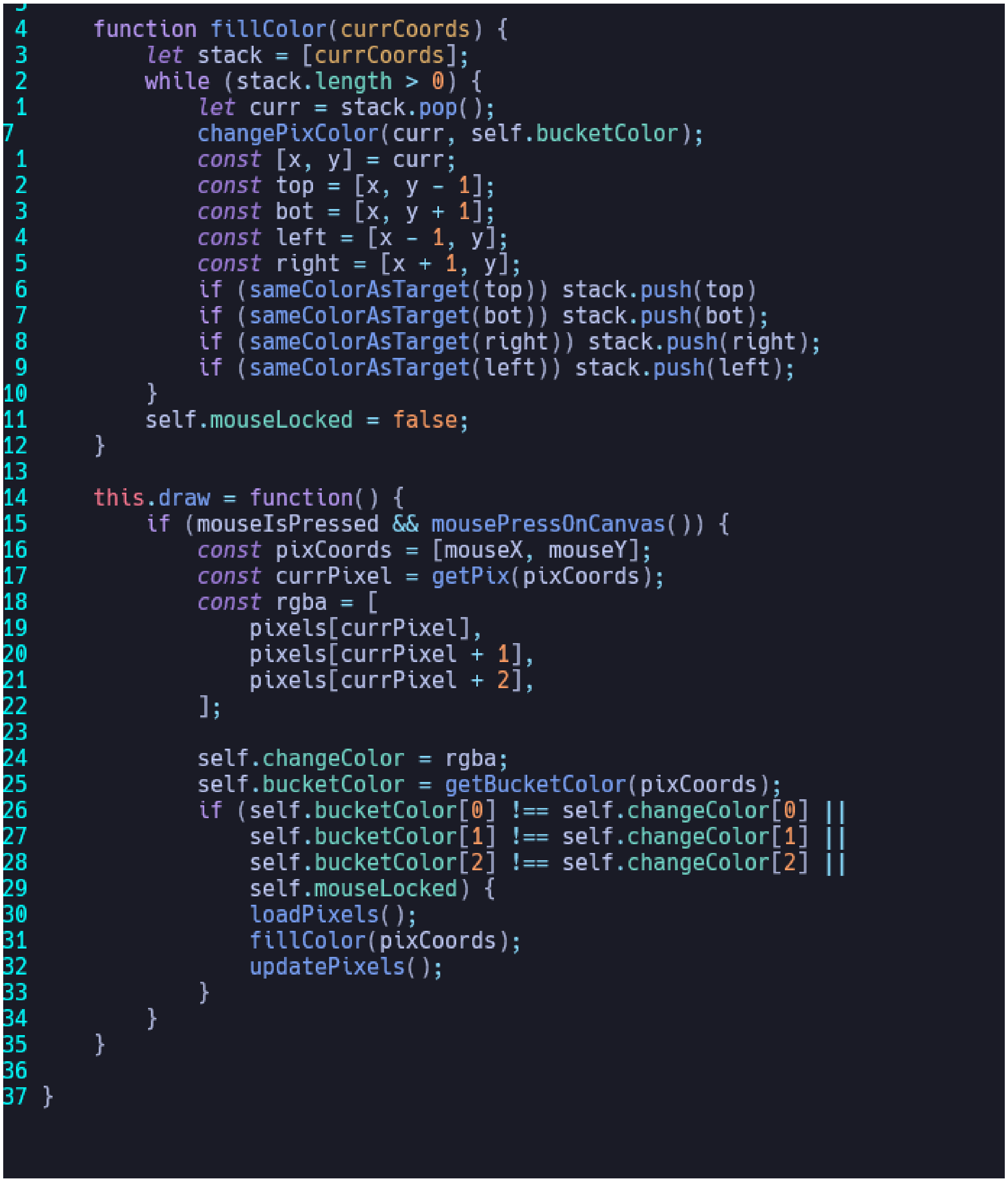
All the pictures below contain code I wrote with zero assistance. Files that have not been included were not written without assistance, and therefore were omitted from these pictures.

FillBucketTool.js

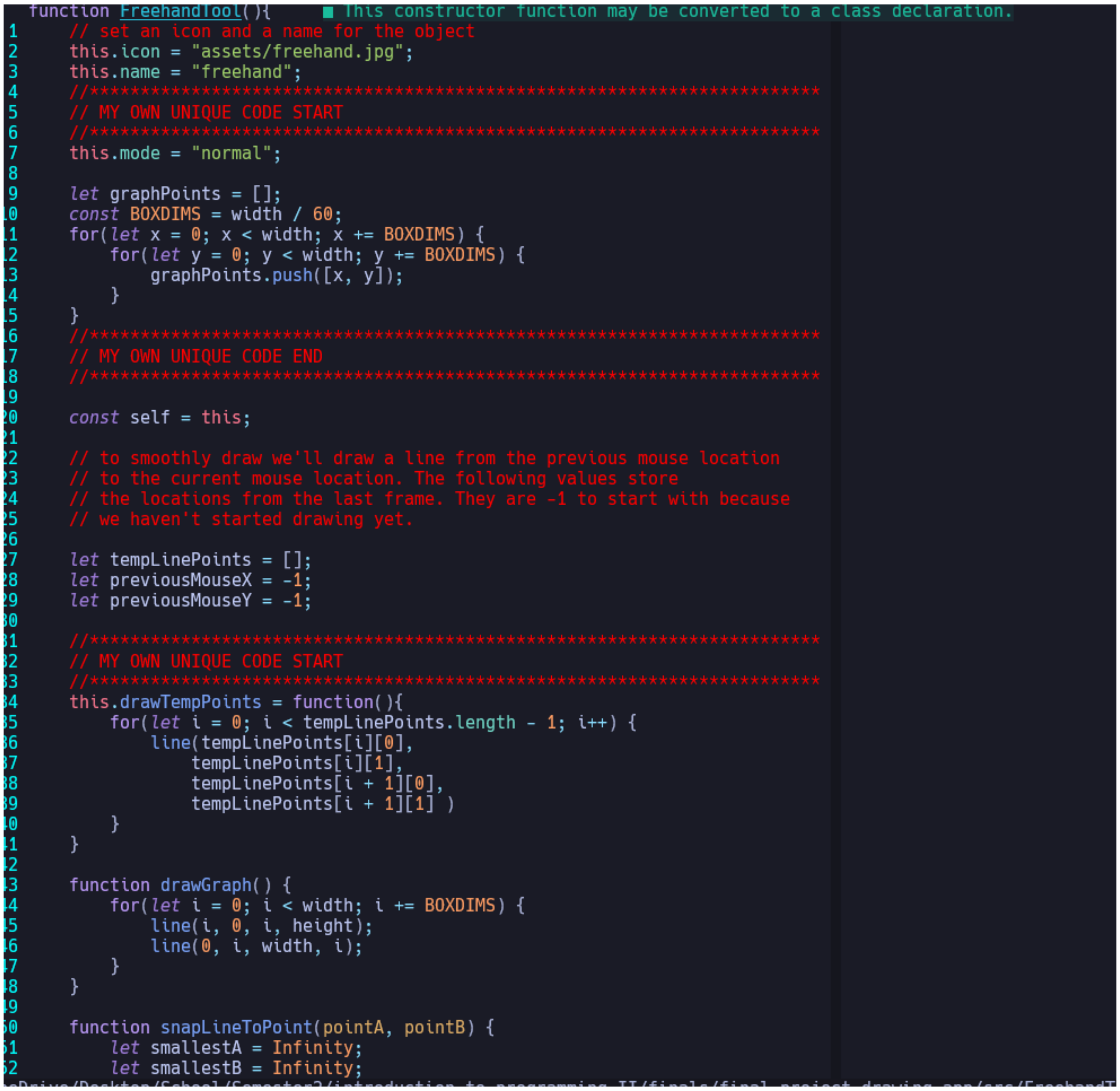


```
// ******************************************************************************
// ALL CODE IN THE FILL BUCKET TOOL HERE WAS WRITTEN BY MYSELF WITH NO HELP
// ******************************************************************************
function FillBucketTool(){     ■ This constructor function may be converted to a class declaration.
    this.name = "fillBucketTool"
    this.icon = "assets/fillBucket.jpg"
    this.changeColor = this.bucketColor = null;
    this.mouseLocked = false;

    const self = this;

    function getPix (coords) {
        return (coords[1] * width + coords[0]) * 4;
    }

    function changePixColor(coords, color) {
        let pix = getPix(coords);
        pixels[pix] = color[0];
        pixels[pix + 1] = color[1];
        pixels[pix + 2] = color[2];
    }

    function sameColorAsTarget(coords) {
        const pix = getPix(coords);
        // We don't care about checking opacity, so we only iterate over the
        // first three values
        if (pixels[pix] !== self.changeColor[0] ||
            pixels[pix + 1] !== self.changeColor[1] ||
            pixels[pix + 2] !== self.changeColor[2]) {
            return false;
        }

        return true;
    }

    function getBucketColor(pixCoords) {
        let [x, y] = pixCoords;
        push();
        noStroke();
        ellipse(x, y, 3);
        loadPixels();
        const pixUnderCursor = getPix(pixCoords);
        const currentColor = [pixels[pixUnderCursor],
            pixels[pixUnderCursor + 1],
            pixels[pixUnderCursor + 2]]
        fill(self.changeColor);
        ellipse(x, y, 6);
        loadPixels();
        pop();
        return currentColor;
    }

    function fillColor(currCoords) {
```

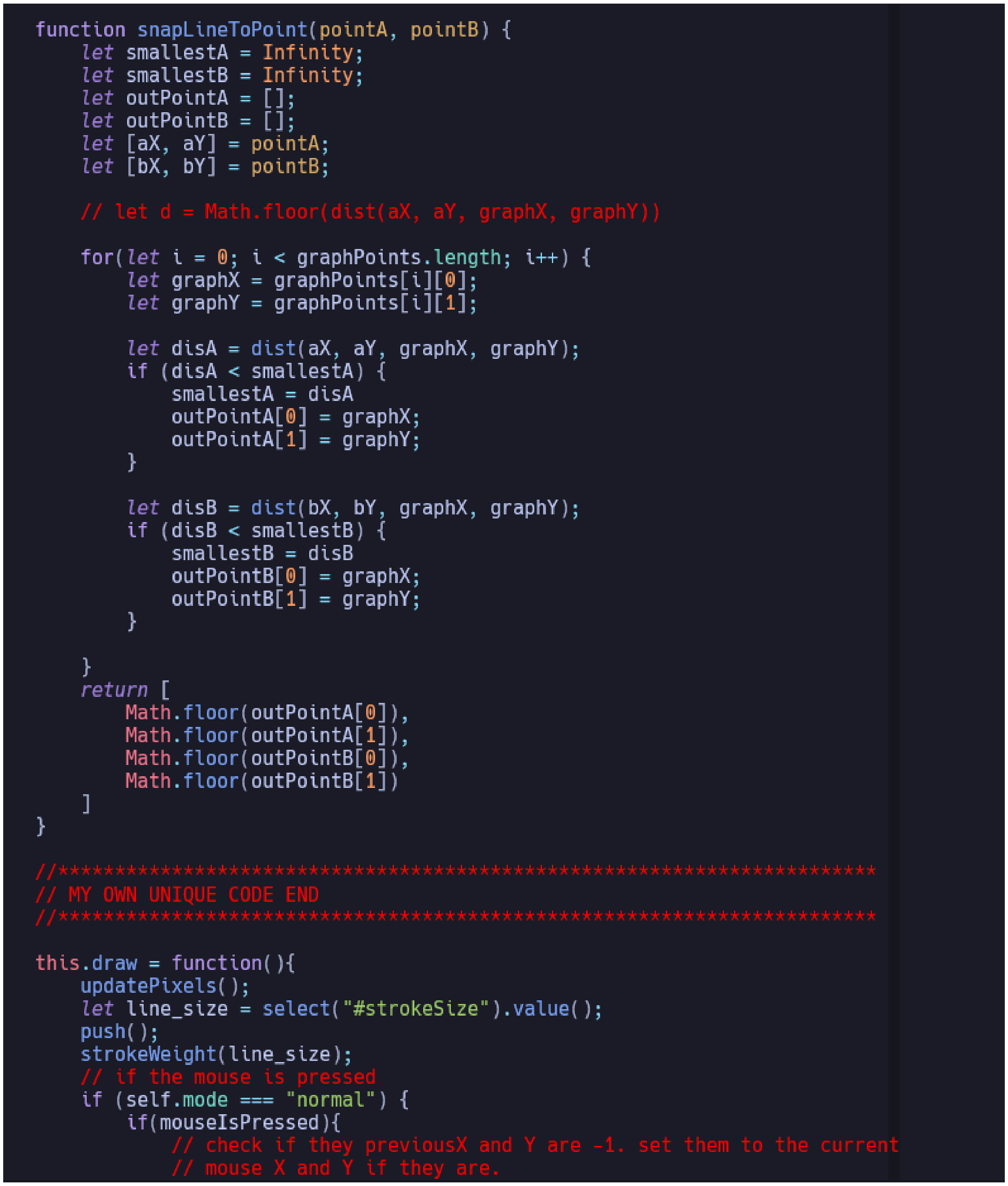


```
    function fillColor(currCoords) {
        let stack = [currCoords];
        while (stack.length > 0) {
            let curr = stack.pop();
            changePixColor(curr, self.bucketColor);
            const [x, y] = curr;
            const top = [x, y - 1];
            const bot = [x, y + 1];
            const left = [x - 1, y];
            const right = [x + 1, y];
            if (sameColorAsTarget(top)) stack.push(top)
            if (sameColorAsTarget(bot)) stack.push(bot);
            if (sameColorAsTarget(right)) stack.push(right);
            if (sameColorAsTarget(left)) stack.push(left);
        }
        self.mouseLocked = false;
    }

    this.draw = function() {
        if (mouseIsPressed && mousePressOnCanvas()) {
            const pixCoords = [mouseX, mouseY];
            const currPixel = getPix(pixCoords);
            const rgba = [
                pixels[currPixel],
                pixels[currPixel + 1],
                pixels[currPixel + 2],
            ];

            self.changeColor = rgba;
            self.bucketColor = getBucketColor(pixCoords);
            if (self.bucketColor[0] !== self.changeColor[0] ||
                self.bucketColor[1] !== self.changeColor[1] ||
                self.bucketColor[2] !== self.changeColor[2] ||
                self.mouseLocked) {
                loadPixels();
                fillColor(pixCoords);
                updatePixels();
            }
        }
    }

}
```

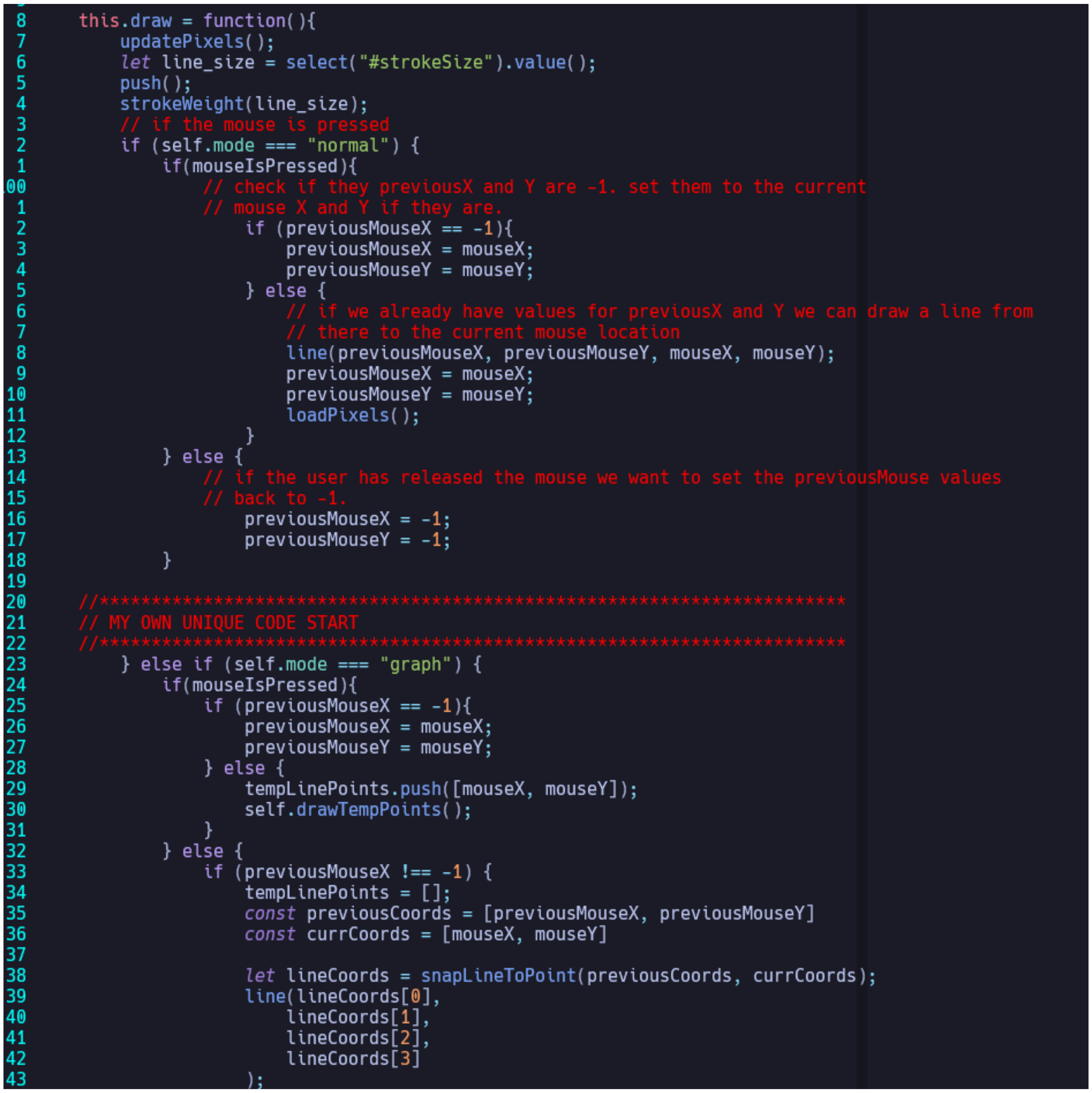
FreeHandTool.js



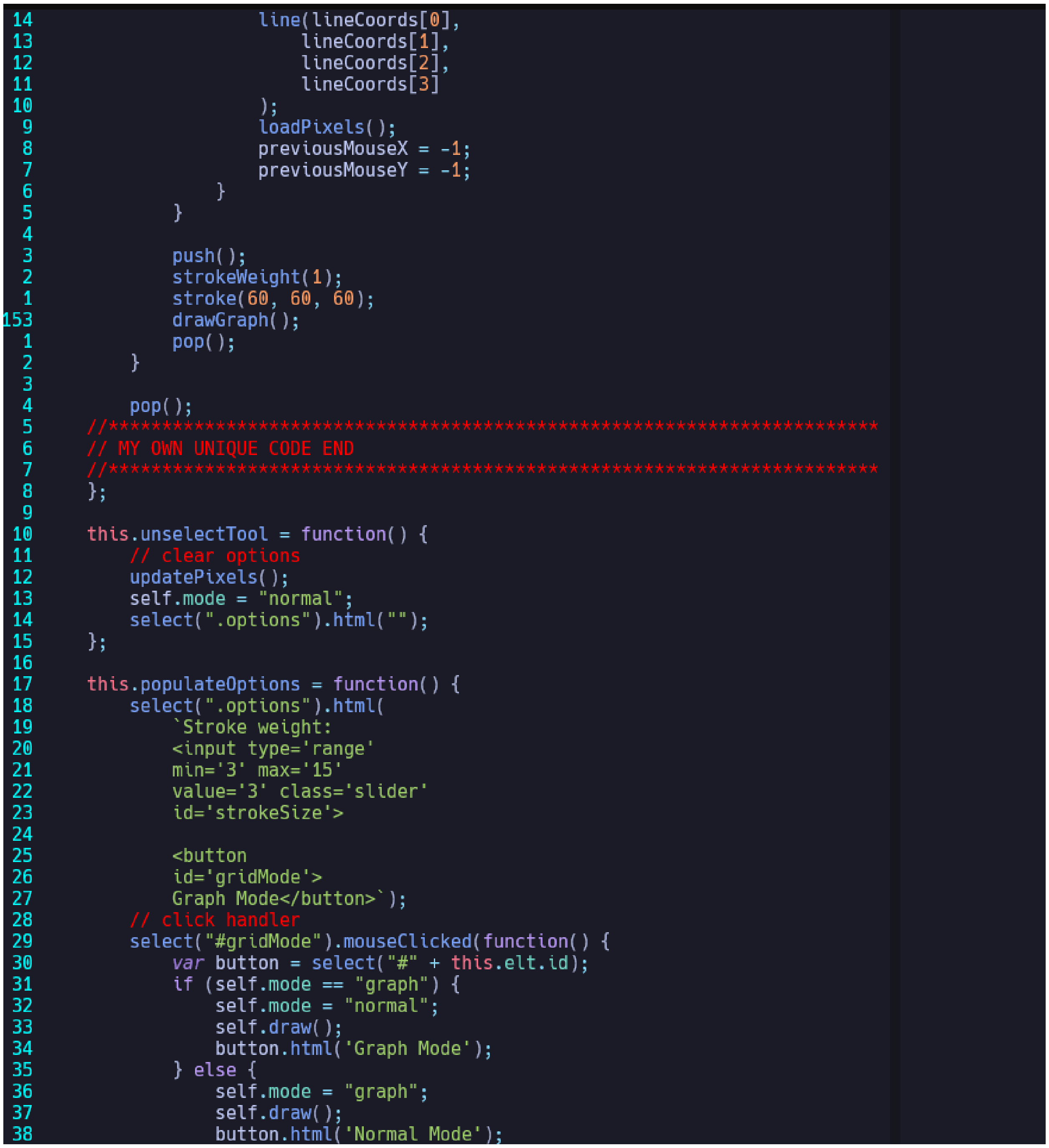
```
function FreehandTool(){      ■ This constructor function may be converted to a class declaration.
    // set an icon and a name for the object
    this.icon = "assets/freehand.jpg";
    this.name = "freehand";
    //*************************************************************************
    // MY OWN UNIQUE CODE START
    //*************************************************************************
    this.mode = "normal";

    let graphPoints = [];
    const BOXDIMS = width / 60;
    for(let x = 0; x < width; x += BOXDIMS) {
        for(let y = 0; y < width; y += BOXDIMS) {
            graphPoints.push([x, y]);
        }
    }
    //*************************************************************************
    // MY OWN UNIQUE CODE END
    //*************************************************************************

    const self = this;

    // to smoothly draw we'll draw a line from the previous mouse location
    // to the current mouse location. The following values store
    // the locations from the last frame. They are -1 to start with because
    // we haven't started drawing yet.

    let tempLinePoints = [];
    let previousMouseX = -1;
    let previousMouseY = -1;

    //*************************************************************************
    // MY OWN UNIQUE CODE START
    //*************************************************************************
    this.drawTempPoints = function(){
        for(let i = 0; i < tempLinePoints.length - 1; i++) {
            line(tempLinePoints[i][0],
                tempLinePoints[i][1],
                tempLinePoints[i + 1][0],
                tempLinePoints[i + 1][1] )
        }
    }

    function drawGraph() {
        for(let i = 0; i < width; i += BOXDIMS) {
            line(i, 0, i, height);
            line(0, i, width, i);
        }
    }

    function snapLineToPoint(pointA, pointB) {
        let smallestA = Infinity;
        let smallestB = Infinity;
eDrive/Desktop/School/Semester2/introduction to programming II/finals/final project drawing app/src/FreehandT
```



```
function snapLineToPoint(pointA, pointB) {
    let smallestA = Infinity;
    let smallestB = Infinity;
    let outPointA = [];
    let outPointB = [];
    let [aX, aY] = pointA;
    let [bX, bY] = pointB;

    // let d = Math.floor(dist(aX, aY, graphX, graphY))

    for(let i = 0; i < graphPoints.length; i++) {
        let graphX = graphPoints[i][0];
        let graphY = graphPoints[i][1];

        let disA = dist(aX, aY, graphX, graphY);
        if (disA < smallestA) {
            smallestA = disA
            outPointA[0] = graphX;
            outPointA[1] = graphY;
        }

        let disB = dist(bX, bY, graphX, graphY);
        if (disB < smallestB) {
            smallestB = disB
            outPointB[0] = graphX;
            outPointB[1] = graphY;
        }

    }
    return [
        Math.floor(outPointA[0]),
        Math.floor(outPointA[1]),
        Math.floor(outPointB[0]),
        Math.floor(outPointB[1])
    ]
}

//**************************************************************************
// MY OWN UNIQUE CODE END
//**************************************************************************

this.draw = function(){
    updatePixels();
    let line_size = select("#strokeSize").value();
    push();
    strokeWeight(line_size);
    // if the mouse is pressed
    if (self.mode === "normal") {
        if(mouseIsPressed){
            // check if they previousX and Y are -1. set them to the current
            // mouse X and Y if they are.
```

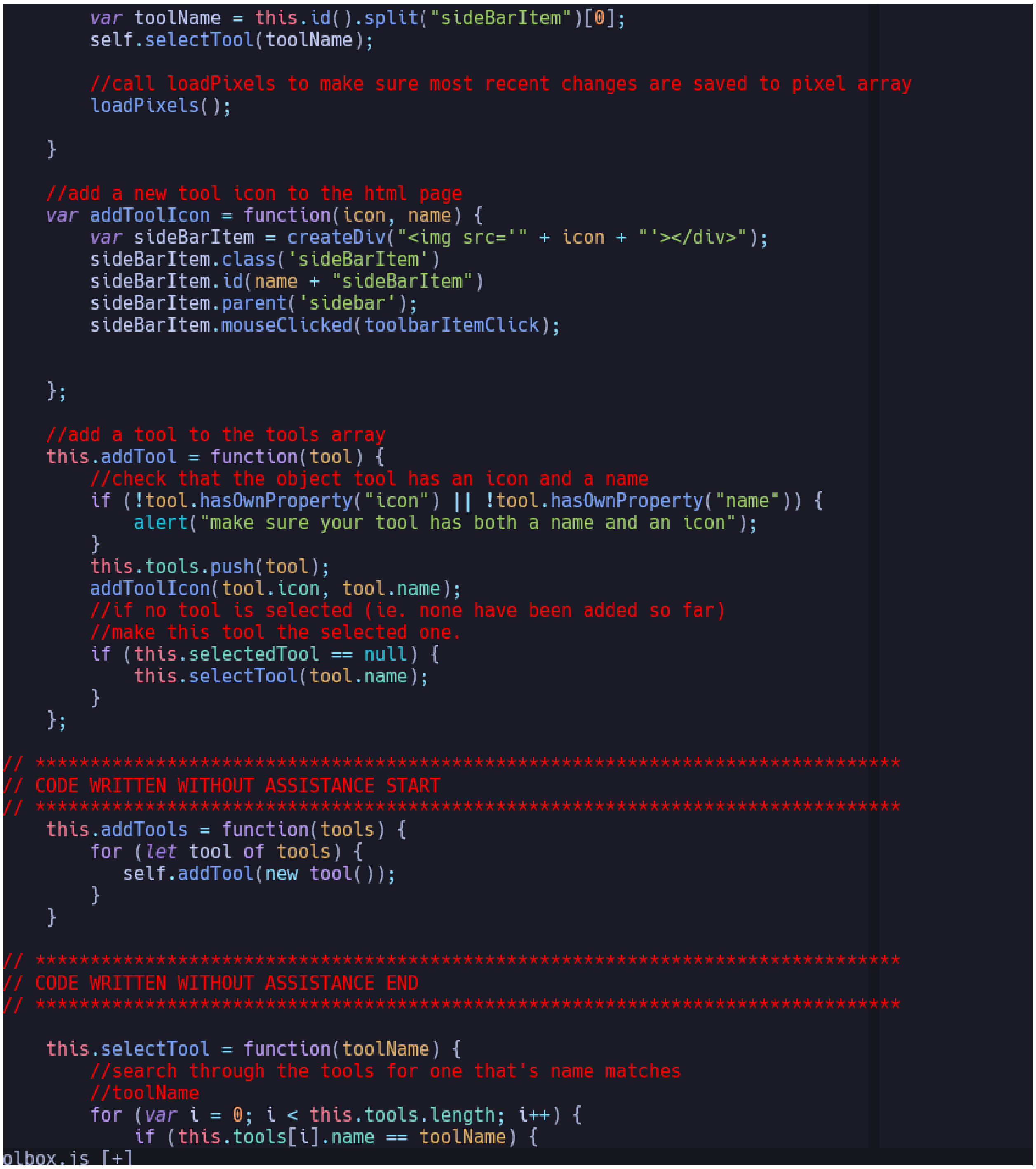


```
    this.draw = function(){
        updatePixels();
        let line_size = select("#strokeSize").value();
        push();
        strokeWeight(line_size);
        // if the mouse is pressed
        if (self.mode === "normal") {
            if(mouseIsPressed){
                // check if they previousX and Y are -1. set them to the current
                // mouse X and Y if they are.
                    if (previousMouseX == -1){
                        previousMouseX = mouseX;
                        previousMouseY = mouseY;
                    } else {
                        // if we already have values for previousX and Y we can draw a line from
                        // there to the current mouse location
                        line(previousMouseX, previousMouseY, mouseX, mouseY);
                        previousMouseX = mouseX;
                        previousMouseY = mouseY;
                        loadPixels();
                    }
            } else {
                // if the user has released the mouse we want to set the previousMouse values
                // back to -1.
                    previousMouseX = -1;
                    previousMouseY = -1;
            }

    //***************************************************************************
    // MY OWN UNIQUE CODE START
    //***************************************************************************
        } else if (self.mode === "graph") {
            if(mouseIsPressed){
                if (previousMouseX == -1){
                    previousMouseX = mouseX;
                    previousMouseY = mouseY;
                } else {
                    tempLinePoints.push([mouseX, mouseY]);
                    self.drawTempPoints();
                }
            } else {
                if (previousMouseX !== -1) {
                    tempLinePoints = [];
                    const previousCoords = [previousMouseX, previousMouseY]
                    const currCoords = [mouseX, mouseY]

                    let lineCoords = snapLineToPoint(previousCoords, currCoords);
                    line(lineCoords[0],
                        lineCoords[1],
                        lineCoords[2],
                        lineCoords[3]
                    );
```



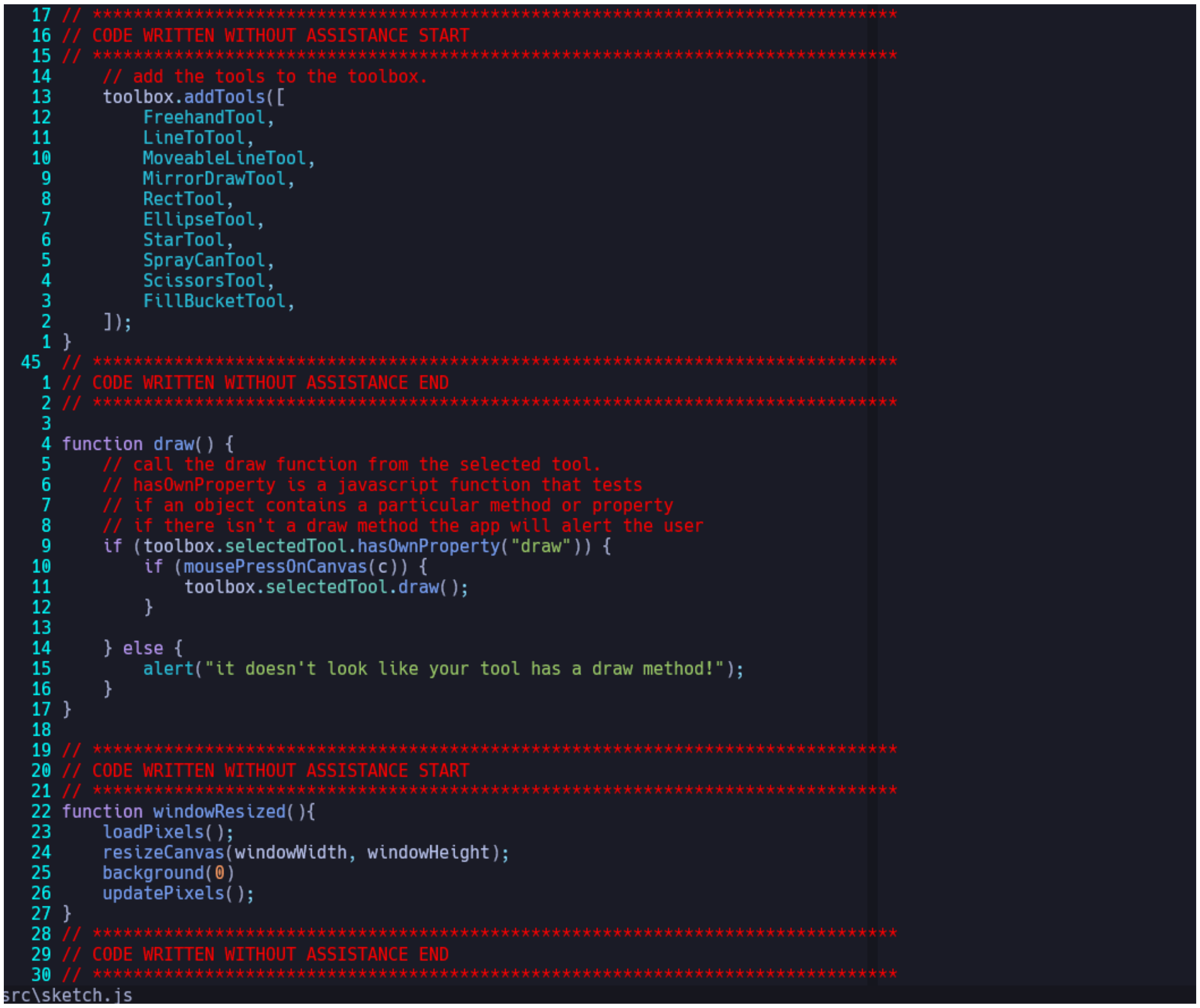
```
                    line(lineCoords[0],
                        lineCoords[1],
                        lineCoords[2],
                        lineCoords[3]
                    );
                    loadPixels();
                    previousMouseX = -1;
                    previousMouseY = -1;
                }
            }

            push();
            strokeWeight(1);
            stroke(60, 60, 60);
            drawGraph();
            pop();
        }

        pop();
    //************************************************************************
    // MY OWN UNIQUE CODE END
    //************************************************************************
    };

    this.unselectTool = function() {
        // clear options
        updatePixels();
        self.mode = "normal";
        select(".options").html("");
    };

    this.populateOptions = function() {
        select(".options").html(
            `Stroke weight:
            <input type='range'
            min='3' max='15'
            value='3' class='slider'
            id='strokeSize'>

            <button
            id='gridMode'>
            Graph Mode</button>`);
        // click handler
        select("#gridMode").mouseClicked(function() {
            var button = select("#" + this.elt.id);
            if (self.mode == "graph") {
                self.mode = "normal";
                self.draw();
                button.html('Graph Mode');
            } else {
                self.mode = "graph";
                self.draw();
                button.html('Normal Mode');
```

Toolbox.js



```
        var toolName = this.id().split("sideBarItem")[0];
        self.selectTool(toolName);

        //call loadPixels to make sure most recent changes are saved to pixel array
        loadPixels();

    }

    //add a new tool icon to the html page
    var addToolIcon = function(icon, name) {
        var sideBarItem = createDiv("<img src='" + icon + "'></div>");
        sideBarItem.class('sideBarItem')
        sideBarItem.id(name + "sideBarItem")
        sideBarItem.parent('sidebar');
        sideBarItem.mouseClicked(toolbarItemClick);


    };

    //add a tool to the tools array
    this.addTool = function(tool) {
        //check that the object tool has an icon and a name
        if (!tool.hasOwnProperty("icon") || !tool.hasOwnProperty("name")) {
            alert("make sure your tool has both a name and an icon");
        }
        this.tools.push(tool);
        addToolIcon(tool.icon, tool.name);
        //if no tool is selected (ie. none have been added so far)
        //make this tool the selected one.
        if (this.selectedTool == null) {
            this.selectTool(tool.name);
        }
    };

// ********************************************************************************
// CODE WRITTEN WITHOUT ASSISTANCE START
// ********************************************************************************
    this.addTools = function(tools) {
        for (let tool of tools) {
           self.addTool(new tool());
        }
    }

// ********************************************************************************
// CODE WRITTEN WITHOUT ASSISTANCE END
// ********************************************************************************

    this.selectTool = function(toolName) {
        //search through the tools for one that's name matches
        //toolName
        for (var i = 0; i < this.tools.length; i++) {
            if (this.tools[i].name == toolName) {
olbox.js [+]
```

Sketch.js



```
// ******************************************************************************
// CODE WRITTEN WITHOUT ASSISTANCE START
// ******************************************************************************
    // add the tools to the toolbox.
    toolbox.addTools([
        FreehandTool,
        LineToTool,
        MoveableLineTool,
        MirrorDrawTool,
        RectTool,
        EllipseTool,
        StarTool,
        SprayCanTool,
        ScissorsTool,
        FillBucketTool,
    ]);
}
// ******************************************************************************
// CODE WRITTEN WITHOUT ASSISTANCE END
// ******************************************************************************

function draw() {
    // call the draw function from the selected tool.
    // hasOwnProperty is a javascript function that tests
    // if an object contains a particular method or property
    // if there isn't a draw method the app will alert the user
    if (toolbox.selectedTool.hasOwnProperty("draw")) {
        if (mousePressOnCanvas(c)) {
            toolbox.selectedTool.draw();
        }

    } else {
        alert("it doesn't look like your tool has a draw method!");
    }
}

// ******************************************************************************
// CODE WRITTEN WITHOUT ASSISTANCE START
// ******************************************************************************
function windowResized(){
    loadPixels();
    resizeCanvas(windowWidth, windowHeight);
    background(0)
    updatePixels();
}
// ******************************************************************************
// CODE WRITTEN WITHOUT ASSISTANCE END
// ******************************************************************************
src\sketch.js
```