Abel Kalamar

/abelkalamar

demo/week03

Past and present

What have I learned this week

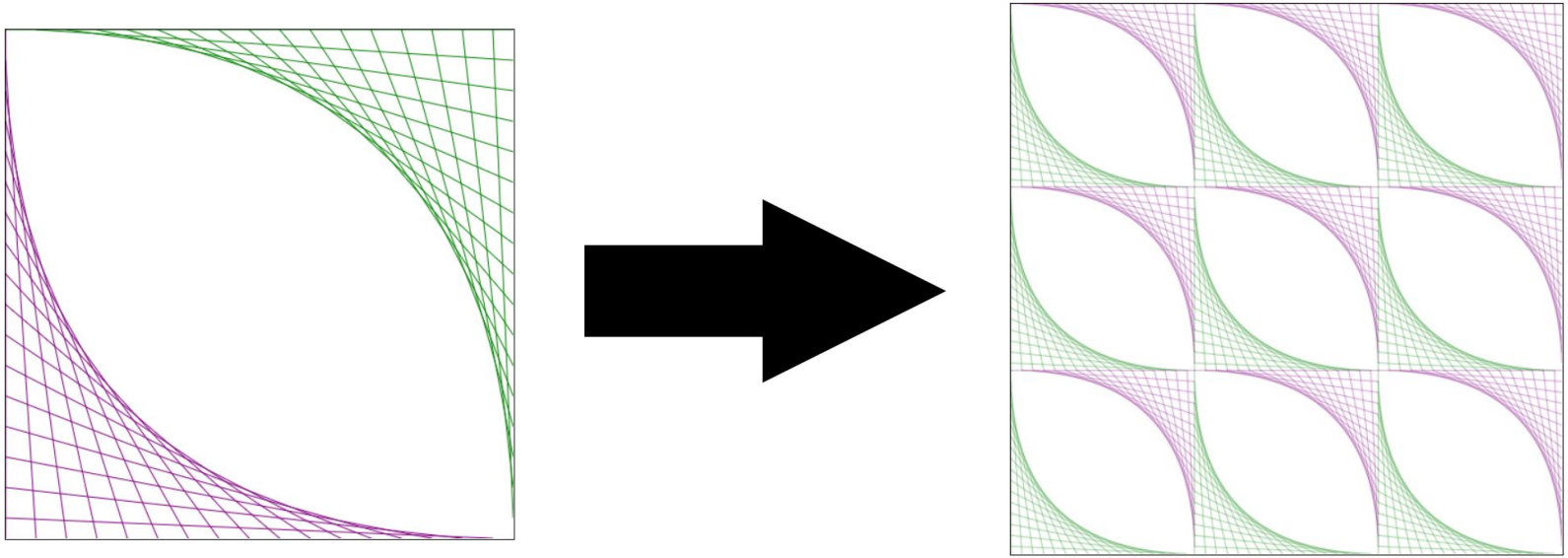
- arrays, strings, objects

- drawing on canvas

Task solution

- Line play quarters

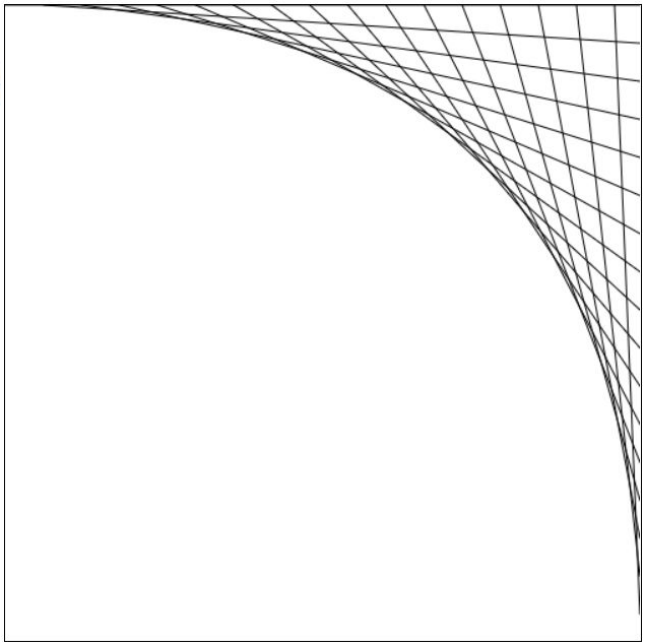
Divide the canvas into 4/16/64 equal parts and repeat the line play pattern in each quarter



Task solution

1. Drawing function:

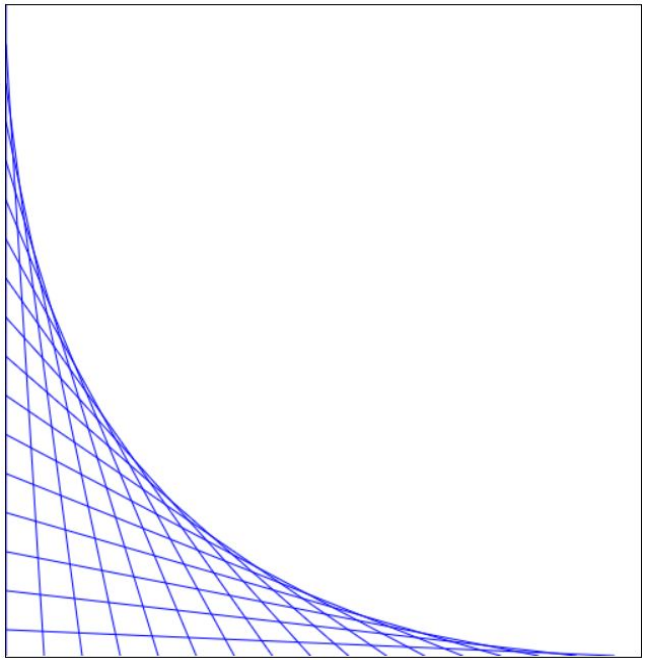
```
function drawLines(lineDistance: number, color: string, direction: number): void {
  ctx.beginPath();
  ctx.strokeStyle = color;
  for (let i: number = 0; i <= canvas.width; i += lineDistance)
    if (direction === 1) {
      ctx.moveTo(i, 0);
      ctx.lineTo(canvas.width, i);
    } else {
      ctx.moveTo(0, i);
      ctx.lineTo(i, canvas.height);
    }
  }
  ctx.stroke();
}
drawLines(30, 'black', 1);
drawLines(30, 'blue', 2);
```



Task solution

1. Drawing function:

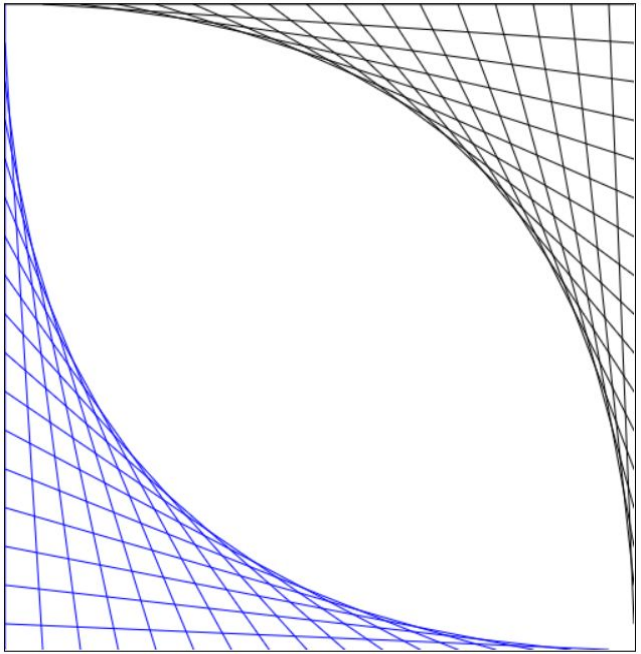
```
function drawLines(lineDistance: number, color: string, direction: number): void {
  ctx.beginPath();
  ctx.strokeStyle = color;
  for (let i: number = 0; i <= canvas.width; i += lineDistance)
    if (direction === 1) {
      ctx.moveTo(i, 0);
      ctx.lineTo(canvas.width, i);
    } else {
      ctx.moveTo(0, i);
      ctx.lineTo(i, canvas.height);
    }
  }
  ctx.stroke();
}
drawLines(30, 'black', 1);
drawLines(30, 'blue', 2);
```



Task solution

1. Drawing function:

```
function drawLines(lineDistance: number, color: string, direction: number): void {
  ctx.beginPath();
  ctx.strokeStyle = color;
  for (let i: number = 0; i <= canvas.width; i += lineDistance)
    if (direction === 1) {
      ctx.moveTo(i, 0);
      ctx.lineTo(canvas.width, i);
    } else {
      ctx.moveTo(0, i);
      ctx.lineTo(i, canvas.height);
    }
  }
  ctx.stroke();
}
drawLines(30, 'black', 1);
drawLines(30, 'blue', 2);
```



Task solution

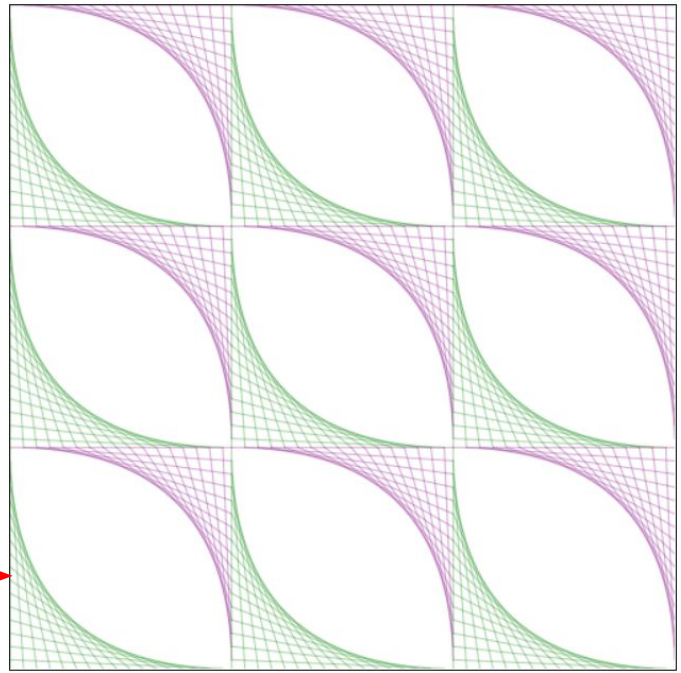
2. Write a multiplier function:

```
translate (x, y);

x: move origo with 'x' pixels
y: move origo with 'y' pixels


scale (a, b);

a: modify drawing horizontal size (if a = 1 -> no effect)
b: modify drawing vertical size (if b = 1 -> no effect)
```

Task solution

2. Write a multiplier function:

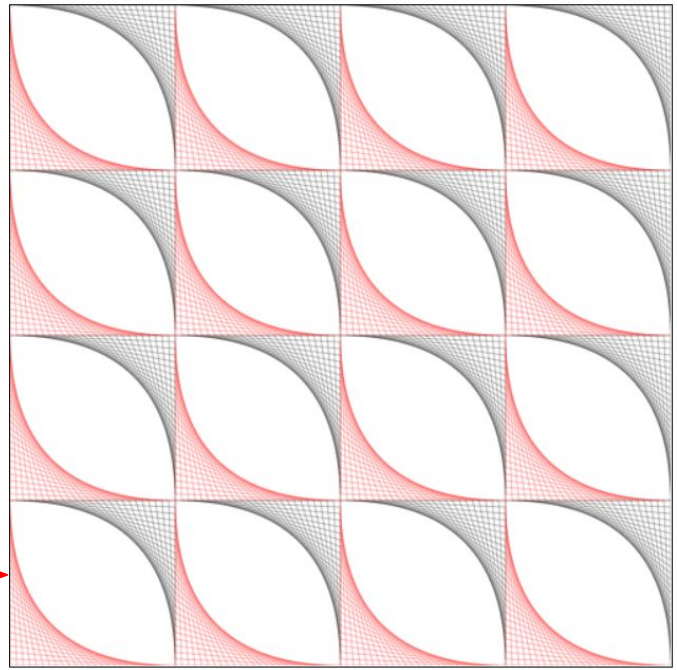
```
function multyLines(num: number, color1: string, color2: string, lineDistance: number) {
  ctx.scale(1 / num, 1 / num);
  let w: number = canvas.width;
  let h: number = canvas.height;
  for (let i: number = 0; i < num; i++) {
    for (let j: number = 0; j < num; j++) {
      drawLines(lineDistance, color1, 1);
      drawLines(lineDistance, color2, 2);
      ctx.translate(w, 0);
    }
    ctx.translate(- num * w, h);
  }
}
multyLines(3, 'purple', 'green', 30);
```



Task solution

2. Write a multiplier function:

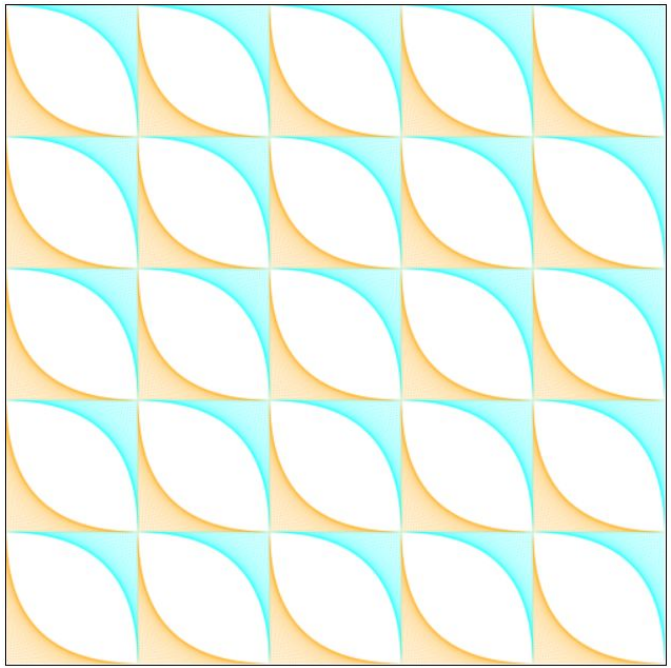
```
function multyLines(num: number, color1: string, color2: string, lineDistance: number) {
  ctx.scale(1 / num, 1 / num);
  let w: number = canvas.width;
  let h: number = canvas.height;
  for (let i: number = 0; i < num; i++) {
    for (let j: number = 0; j < num; j++) {
      drawLines(lineDistance, color1, 1);
      drawLines(lineDistance, color2, 2);
      ctx.translate(w, 0);
    }
    ctx.translate(- num * w, h);
  }
}
multyLines(4, 'black', 'red', 20);
```



Task solution

2. Write a multiplier function:

```
function multyLines(num: number, color1: string, color2: string, lineDistance: number) {
  ctx.scale(1 / num, 1 / num);
  let w: number = canvas.width;
  let h: number = canvas.height;
  for (let i: number = 0; i < num; i++) {
    for (let j: number = 0; j < num; j++) {
      drawLines(lineDistance, color1, 1);
      drawLines(lineDistance, color2, 2);
      ctx.translate(w, 0);
    }
    ctx.translate(- num * w, h);
  }
}
multyLines(5, 'aqua', 'orange', 10);
```



Thank you for your attention!