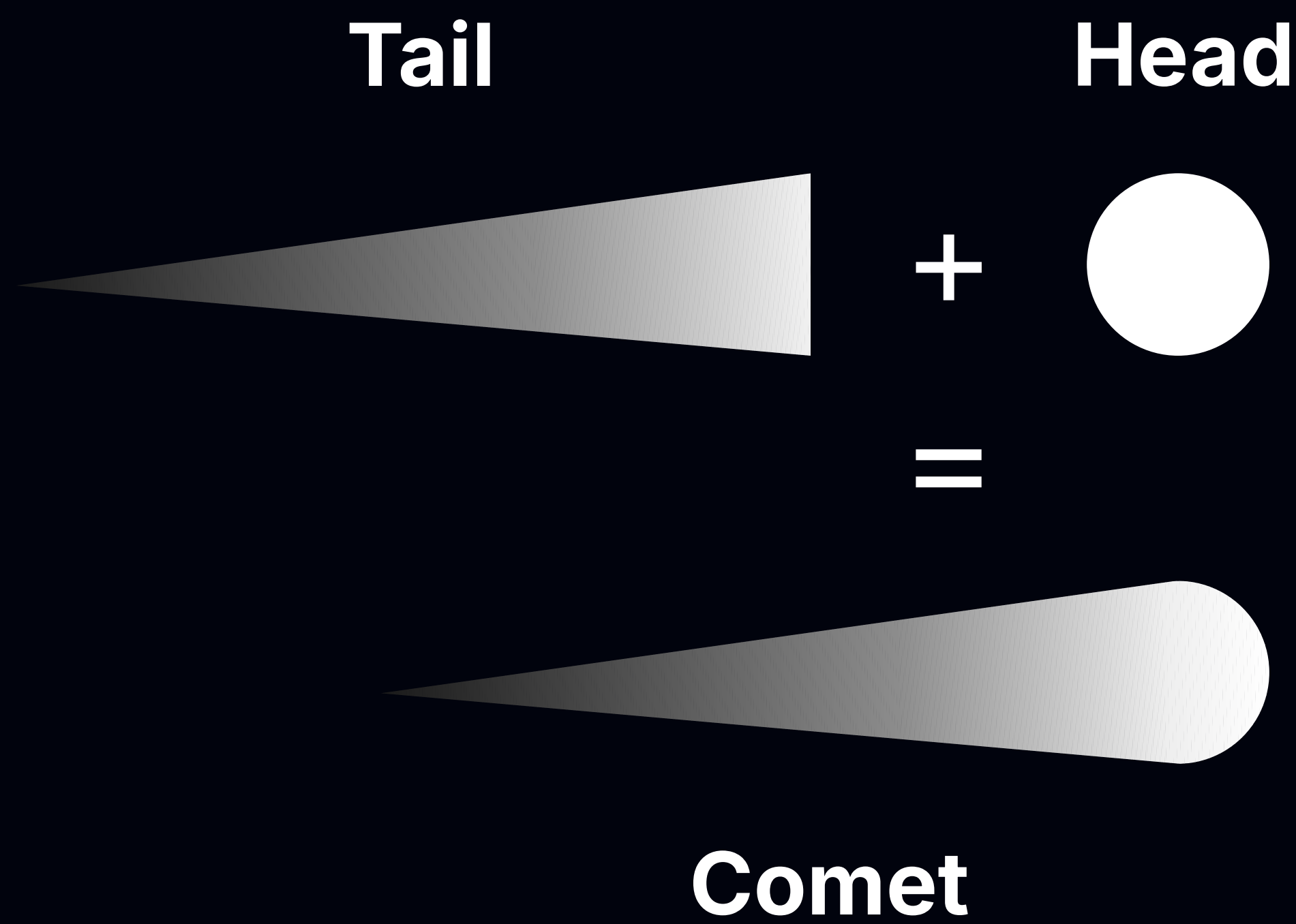


# Meteor shower



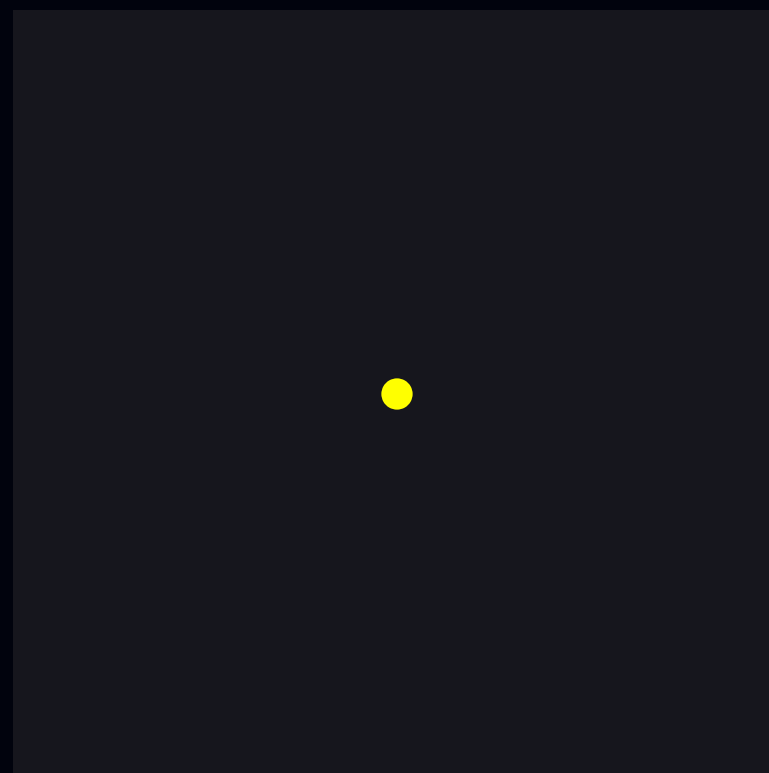
# Designing the comet



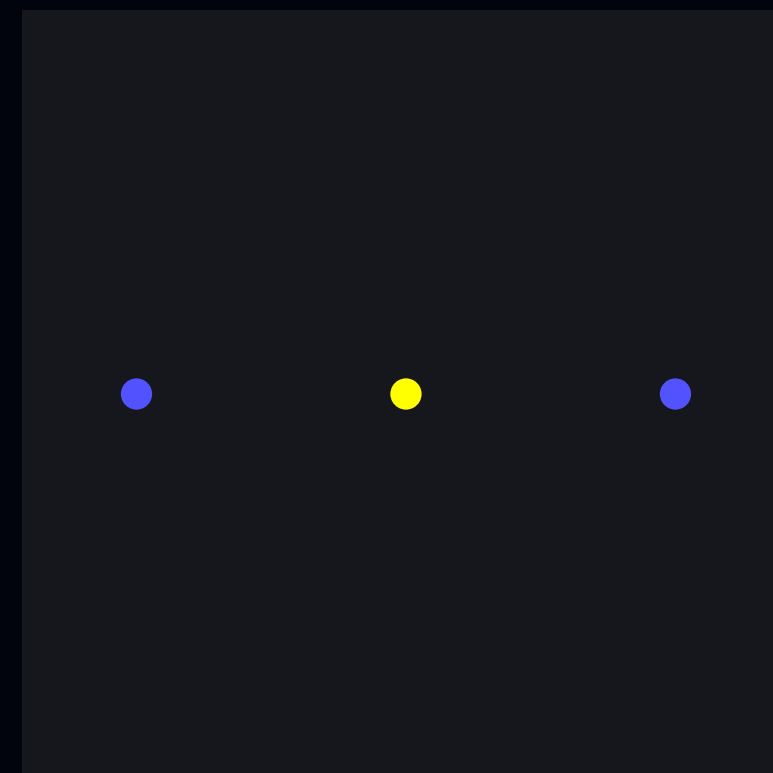
# Designing the comet

To draw the comet, we will use the **canvas** in HTML.

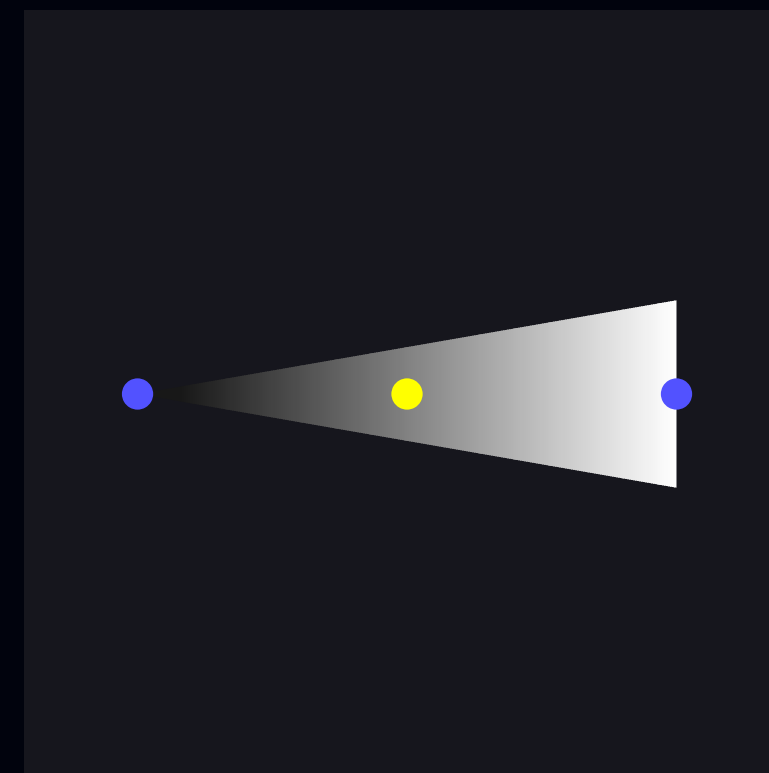
The steps to follow:



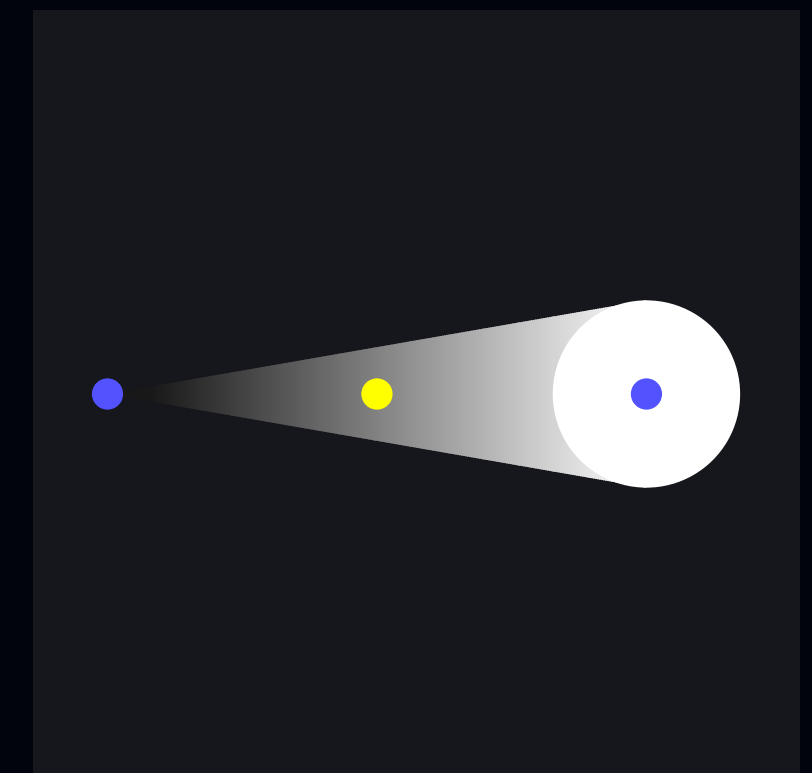
Find the center



Locate the head  
and tail



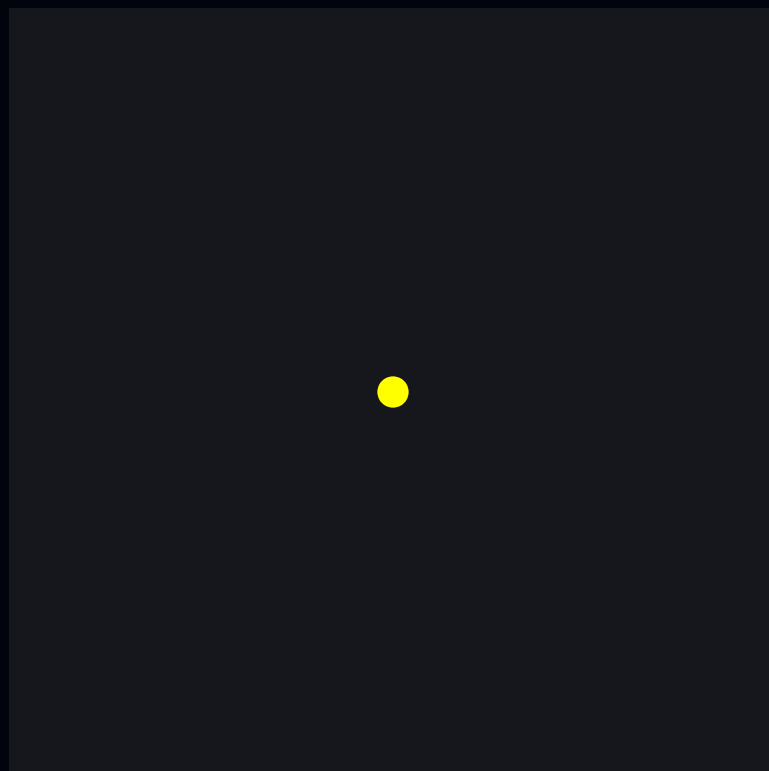
Draw the tail



Add the head

# Designing the comet

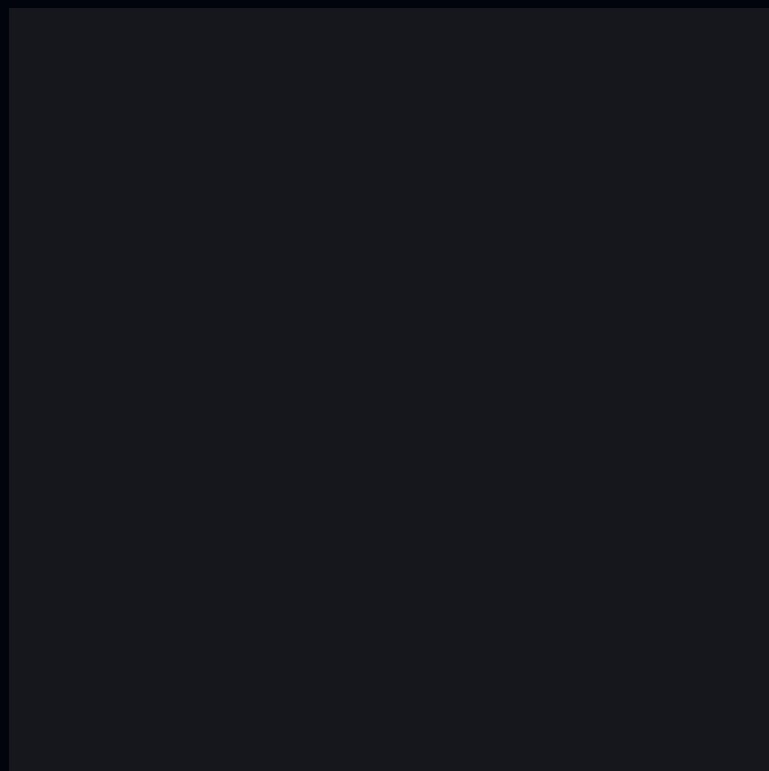
Find the center



```
1  drawComet() {  
2    this.context.clearRect(0, 0, this.width, this.height);  
3  
4    const centerX = this.width / 2;  
5    const centerY = this.height / 2;  
6  
7    // Draw the center point  
8    this.context.beginPath();  
9    this.context.arc(centerX, centerY, 8, 0, 2 * Math.PI);  
10   this.context.fillStyle = "yellow";  
11   this.context.fill();  
12 }
```

# Designing the comet

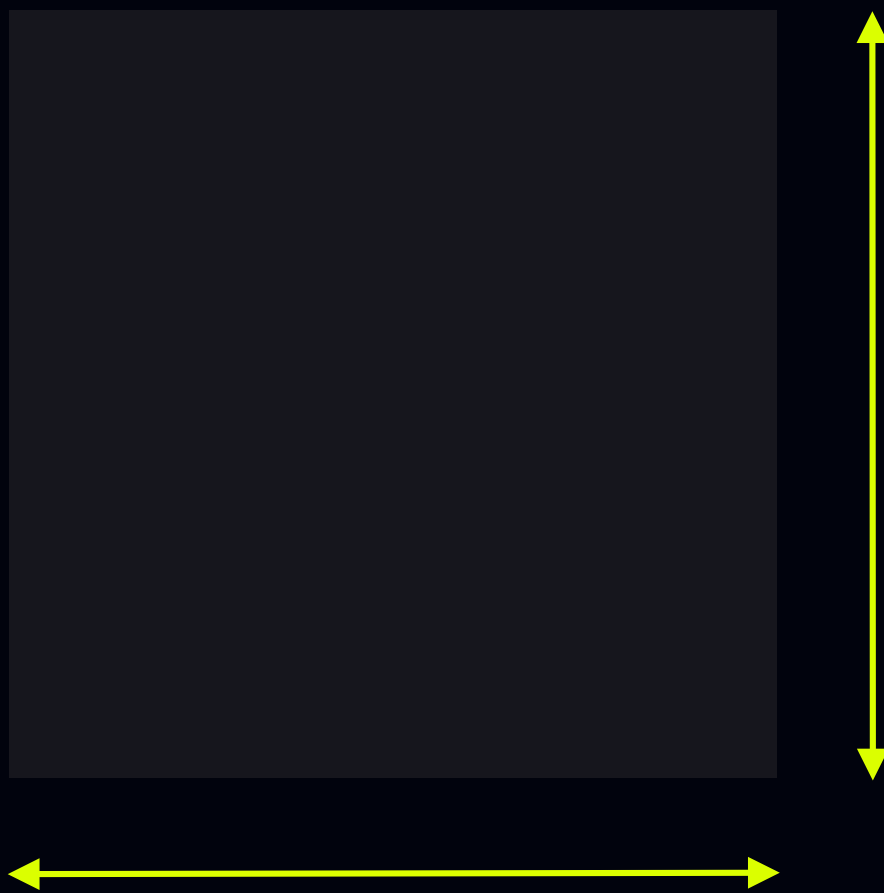
Find the center



```
1  drawComet() {  
2    this.context.clearRect(0, 0, this.width, this.height);  
3  
4    const centerX = this.width / 2;  
5    const centerY = this.height / 2;  
6  
7    // Draw the center point  
8    this.context.beginPath();  
9    this.context.arc(centerX, centerY, 8, 0, 2 * Math.PI);  
10   this.context.fillStyle = "yellow";  
11   this.context.fill();  
12 }
```

# Designing the comet

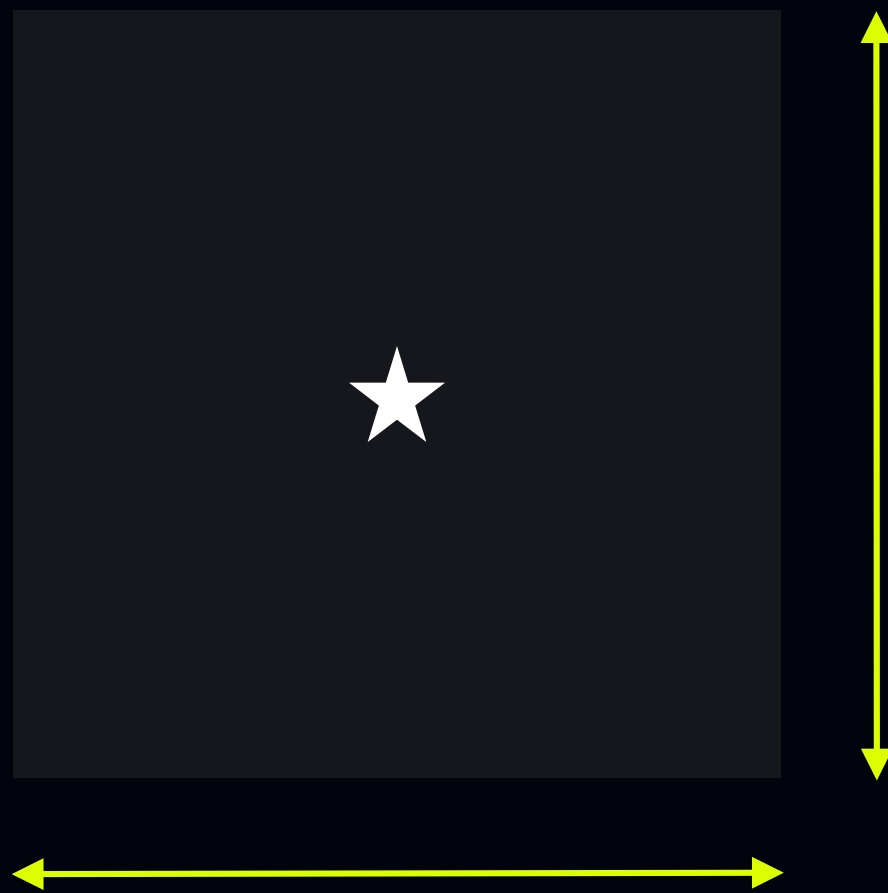
Find the center



```
1  drawComet() {  
2      this.context.clearRect(0, 0, this.width, this.height);  
3  
4      const centerX = this.width / 2;  
5      const centerY = this.height / 2;  
6  
7      // Draw the center point  
8      this.context.beginPath();  
9      this.context.arc(centerX, centerY, 8, 0, 2 * Math.PI);  
10     this.context.fillStyle = "yellow";  
11     this.context.fill();  
12 }
```

# Designing the comet

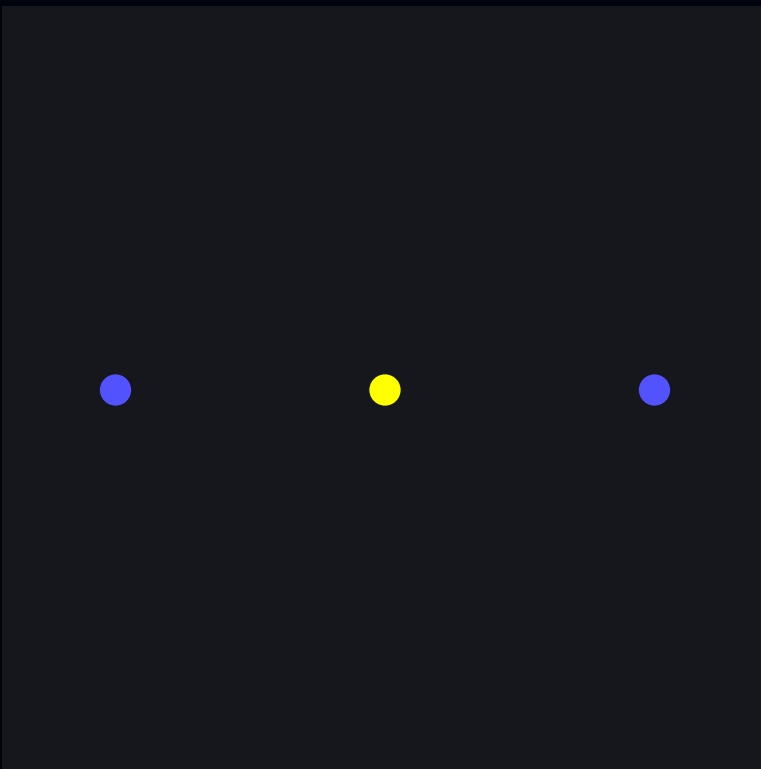
Find the center



```
1  drawComet() {  
2      this.context.clearRect(0, 0, this.width, this.height);  
3  
4      const centerX = this.width / 2;  
5      const centerY = this.height / 2;  
6  
7      // Draw the center point  
8      this.context.beginPath();  
9      this.context.arc(centerX, centerY, 8, 0, 2 * Math.PI);  
10     this.context.fillStyle = "yellow";  
11     this.context.fill();  
12 }
```

# Designing the comet

Locate the head and tail

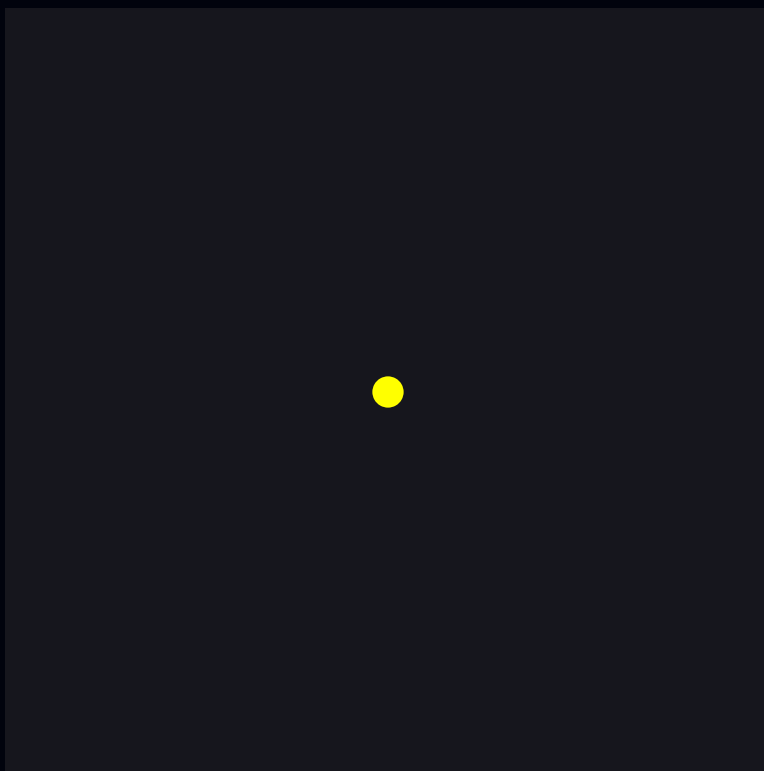


```
1  drawComet() {  
2      ...  
3  
4      const scaledLength = this.comet.length * this.scale;  
5      const scaledLineWidth = this.comet.lineWidth * this.scale;  
6  
7      const headX = centerX + (scaledLength / 2) * Math.cos(this.comet.angle);  
8      const headY = centerY + (scaledLength / 2) * Math.sin(this.comet.angle);  
9  
10     // Draw the point at the head  
11     this.context.beginPath();  
12     this.context.arc(headX, headY, 8, 0, 2 * Math.PI);  
13     this.context.fillStyle = "blue";  
14     this.context.fill();  
15  
16     const tailX = centerX - (scaledLength / 2) * Math.cos(this.comet.angle);  
17     const tailY = centerY - (scaledLength / 2) * Math.sin(this.comet.angle);  
18  
19     // Draw the point at the tail end  
20     this.context.beginPath();  
21     this.context.arc(tailX, tailY, 8, 0, 2 * Math.PI);  
22     this.context.fillStyle = "blue";  
23     this.context.fill();  
24 }
```



# Designing the comet

Locate the head and tail



```
1  drawComet() {  
2    ...  
3  
4    const scaledLength = this.comet.length * this.scale;  
5    const scaledLineWidth = this.comet.lineWidth * this.scale;  
6  
7    const headX = centerX + (scaledLength / 2)  
8    const headY = centerY + (scaledLength / 2)  
9  
10   // Draw the point at the head  
11   this.context.beginPath();  
12   this.context.arc(headX, headY, 8, 0, 2 * Math.PI);  
13   this.context.fillStyle = "blue";  
14   this.context.fill();  
15  
16   const tailX = centerX - (scaledLength / 2)  
17   const tailY = centerY - (scaledLength / 2)  
18  
19   // Draw the point at the tail end  
20   this.context.beginPath();  
21   this.context.arc(tailX, tailY, 8, 0, 2 * Math.PI);  
22   this.context.fillStyle = "blue";  
23   this.context.fill();  
24 },
```

# Designing the comet

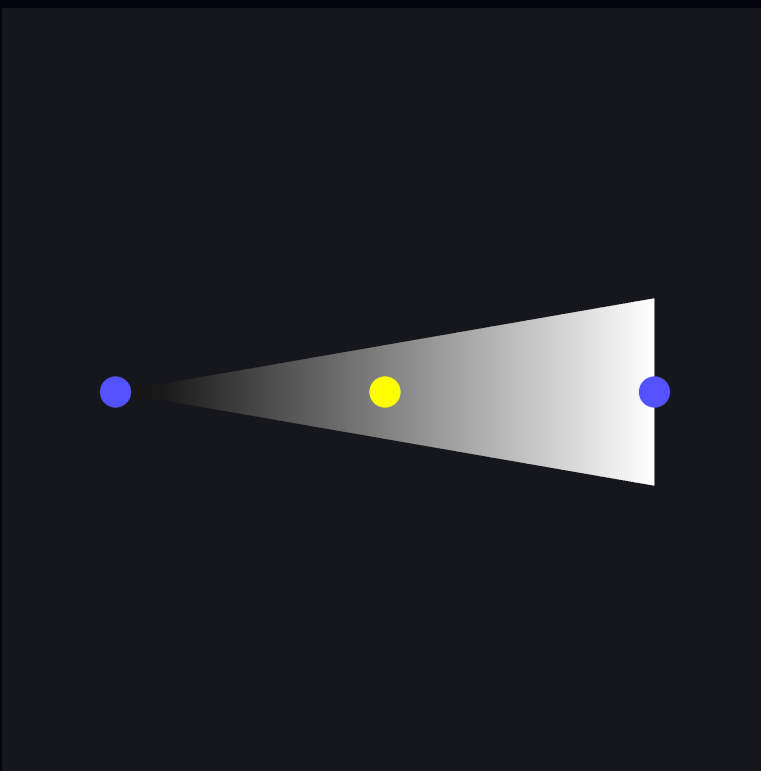
Locate the head and tail



```
1  drawComet() {  
2    ...  
3  
4    const scaledLength = this.comet.length * this.scale;  
5    const scaledLineWidth = this.comet.lineWidth * this.scale;  
6  
7    const headX = centerX + (scaledLength / 2) * Math.cos(this.comet.angle);  
8    const headY = centerY + (scaledLength / 2) * Math.sin(this.comet.angle);  
9  
10   // Draw the point at the head  
11   this.context.beginPath();  
12   this.context.arc(headX, headY, 8, 0, 2 * Math.PI);  
13   this.context.fillStyle = "blue";  
14   this.context.fill();  
15  
16   const tailX = centerX - (scaledLength / 2)  
17   const tailY = centerY - (scaledLength / 2)  
18  
19   // Draw the point at the tail end  
20   this.context.beginPath();  
21   this.context.arc(tailX, tailY, 8, 0, 2 * Math.PI);  
22   this.context.fillStyle = "blue";  
23   this.context.fill();  
24 },
```

# Designing the comet

## Draw the tail



```
1  drawComet() {  
2      ...  
3  
4      // 1. Draw the tail  
5      const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);  
6      tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");  
7      tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");  
8  
9      const perpX = -Math.sin(this.comet.angle);  
10     const perpY = Math.cos(this.comet.angle);  
11     const headBase1X = headX + (scaledLineWidth / 2) * perpX;  
12     const headBase1Y = headY + (scaledLineWidth / 2) * perpY;  
13     const headBase2X = headX - (scaledLineWidth / 2) * perpX;  
14     const headBase2Y = headY - (scaledLineWidth / 2) * perpY;  
15  
16     this.context.beginPath();  
17     this.context.moveTo(headBase1X, headBase1Y);  
18     this.context.lineTo(headBase2X, headBase2Y);  
19     this.context.lineTo(tailX, tailY);  
20     this.context.closePath();  
21     this.context.fillStyle = tailGradient;  
22     this.context.fill();  
23 }
```

# Designing the comet

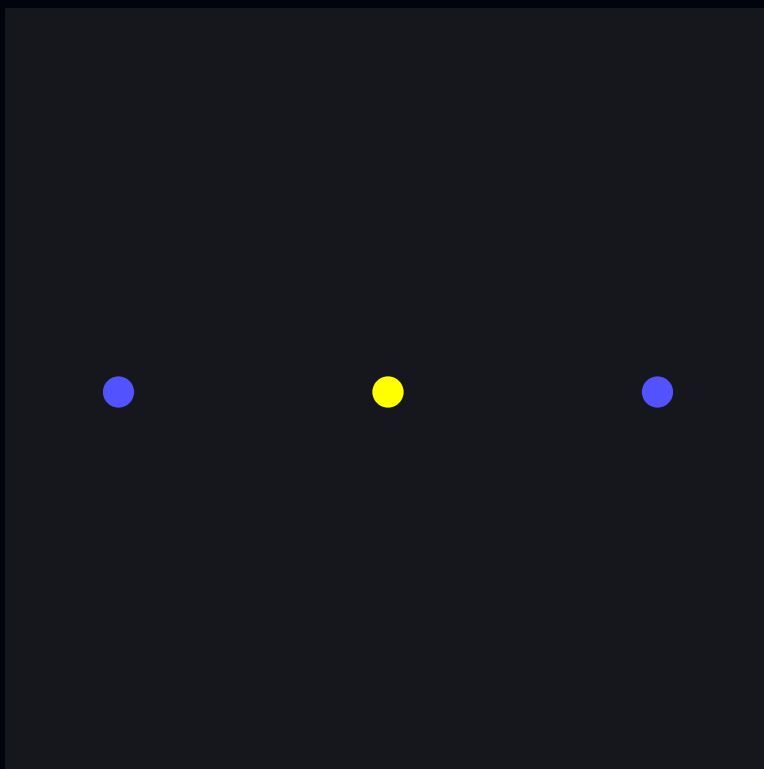
## Draw the tail



```
1  drawComet() {  
2    ...  
3  
4    // 1. Draw the tail  
5    const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);  
6    tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");  
7    tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");  
8  
9    const perpX = -Math.sin(this.comet.angle);  
10   const perpY = Math.cos(this.comet.angle);  
11   const headBase1X = headX + (scaledLineWidth / 2) * perpX;  
12   const headBase1Y = headY + (scaledLineWidth / 2) * perpY;  
13   const headBase2X = headX - (scaledLineWidth / 2) * perpX;  
14   const headBase2Y = headY - (scaledLineWidth / 2) * perpY;  
15  
16   this.context.beginPath();  
17   this.context.moveTo(headBase1X, headBase1Y);  
18   this.context.lineTo(headBase2X, headBase2Y);  
19   this.context.lineTo(tailX, tailY);  
20   this.context.closePath();  
21   this.context.fillStyle = tailGradient;  
22   this.context.fill();  
23 }
```

# Designing the comet

## Draw the tail

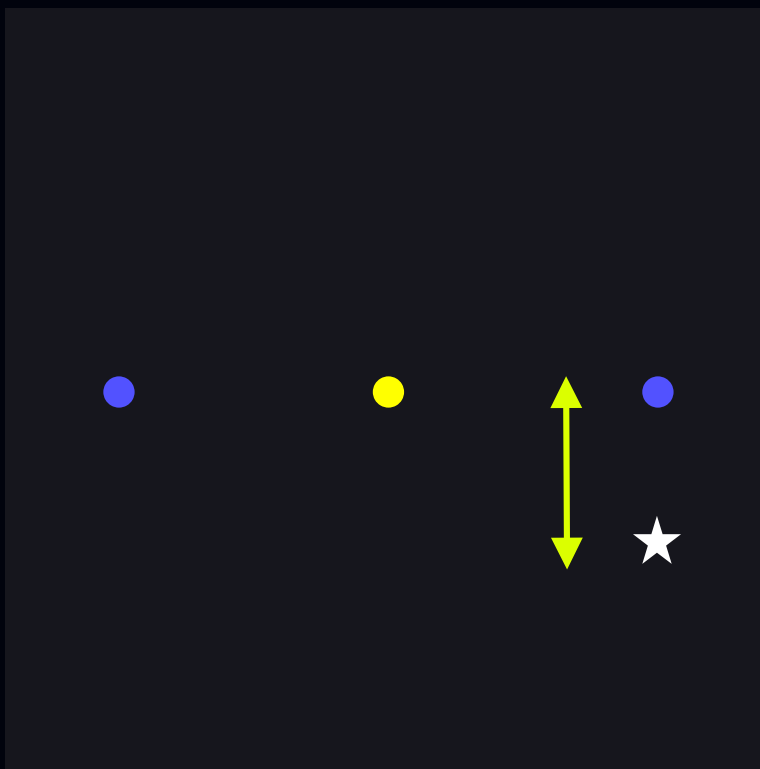


$$\text{perpX} = \sin(0) = 0$$
$$\text{perpY} = \cos(0) = 1$$

```
1  drawComet() {  
2      ...  
3  
4      // 1. Draw the tail  
5      const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);  
6      tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");  
7      tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");  
8  
9      const perpX = -Math.sin(this.comet.angle);  
10     const perpY = Math.cos(this.comet.angle);  
11  
12     const headBase1X = headX + (scaledLineWidth / 2) * perpX;  
13     const headBase1Y = headY + (scaledLineWidth / 2) * perpY;  
14     const headBase2X = headX - (scaledLineWidth / 2) * perpX;  
15     const headBase2Y = headY - (scaledLineWidth / 2) * perpY;  
16  
17     this.context.beginPath();  
18     this.context.moveTo(headBase1X, headBase1Y);  
19     this.context.lineTo(headBase2X, headBase2Y);  
20     this.context.lineTo(tailX, tailY);  
21     this.context.closePath();  
22     this.context.fillStyle = tailGradient;  
23     this.context.fill();  
24 }
```

# Designing the comet

## Draw the tail

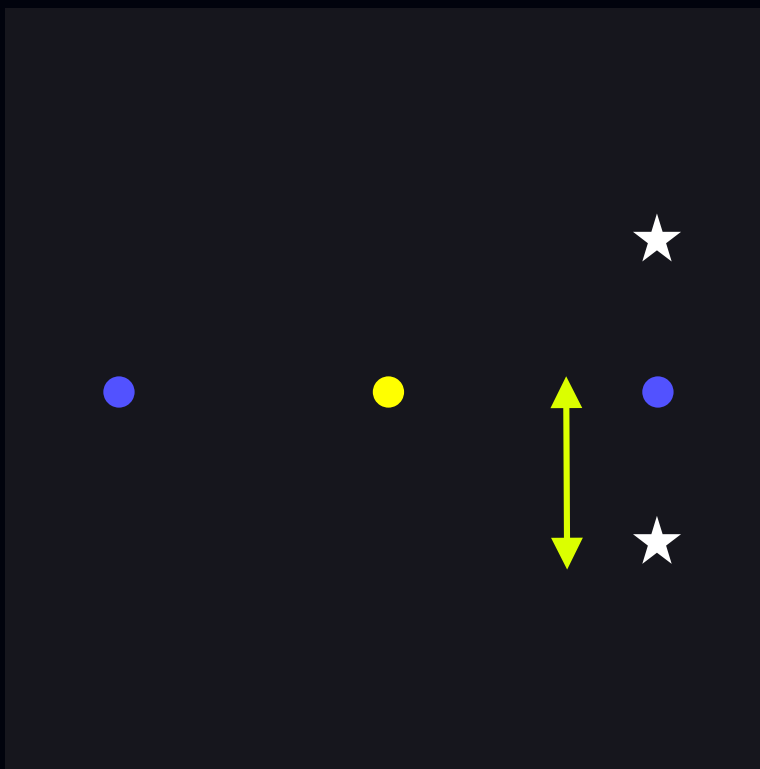


$$\text{perpX} = \sin(0) = 0$$
$$\text{perpY} = \cos(0) = 1$$

```
1  drawComet() {  
2      ...  
3  
4      // 1. Draw the tail  
5      const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);  
6      tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");  
7      tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");  
8  
9      const perpX = -Math.sin(this.comet.angle);  
10     const perpY = Math.cos(this.comet.angle);  
11     const headBase1X = headX + (scaledLineWidth / 2) * perpX;  
12     const headBase1Y = headY + (scaledLineWidth / 2) * perpY;  
13     const headBase2X = headX - (scaledLineWidth / 2) * perpX;  
14     const headBase2Y = headY - (scaledLineWidth / 2) * perpY;  
15  
16     this.context.beginPath();  
17     this.context.moveTo(headBase1X, headBase1Y);  
18     this.context.lineTo(headBase2X, headBase2Y);  
19     this.context.lineTo(tailX, tailY);  
20     this.context.closePath();  
21     this.context.fillStyle = tailGradient;  
22     this.context.fill();  
23 }
```

# Designing the comet

## Draw the tail



$$\text{perpX} = \sin(0) = 0$$
$$\text{perpY} = \cos(0) = 1$$

```
1  drawComet() {
2      ...
3
4      // 1. Draw the tail
5      const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);
6      tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");
7      tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");
8
9      const perpX = -Math.sin(this.comet.angle);
10     const perpY = Math.cos(this.comet.angle);
11     const headBase1X = headX + (scaledLineWidth / 2) * perpX;
12     const headBase1Y = headY + (scaledLineWidth / 2) * perpY;
13     const headBase2X = headX - (scaledLineWidth / 2) * perpX;
14     const headBase2Y = headY - (scaledLineWidth / 2) * perpY;
15
16     this.context.beginPath();
17     this.context.moveTo(headBase1X, headBase1Y);
18     this.context.lineTo(headBase2X, headBase2Y);
19     this.context.lineTo(tailX, tailY);
20     this.context.closePath();
21     this.context.fillStyle = tailGradient;
22     this.context.fill();
23 }
```

# Designing the comet

## Draw the tail

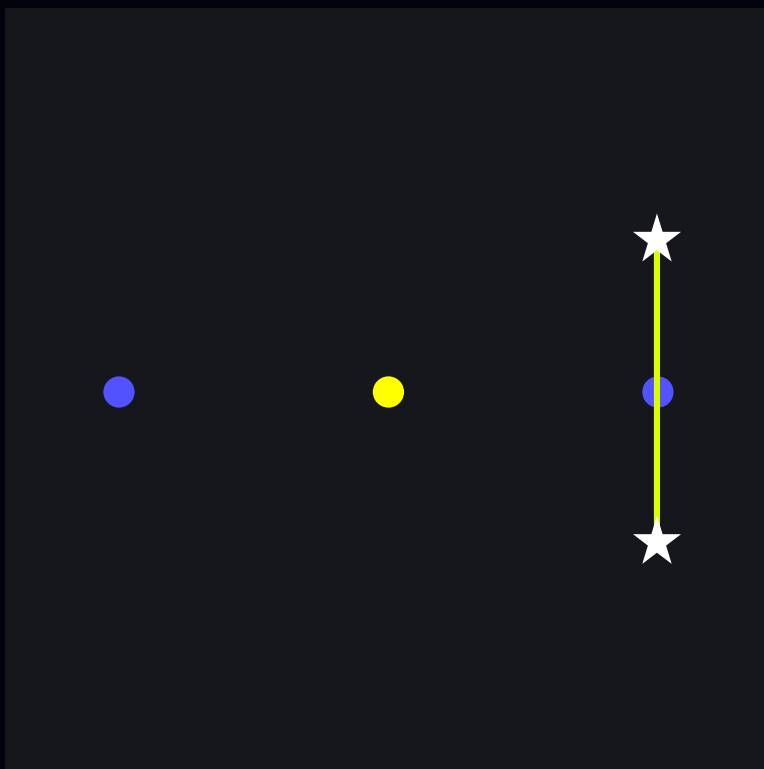


```
1  drawComet() {
2      ...
3
4      // 1. Draw the tail
5      const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);
6      tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");
7      tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");
8
9      const perpX = -Math.sin(this.comet.angle);
10     const perpY = Math.cos(this.comet.angle);
11     const headBase1X = headX + (scaledLineWidth / 2) * perpX;
12     const headBase1Y = headY + (scaledLineWidth / 2) * perpY;
13     const headBase2X = headX - (scaledLineWidth / 2) * perpX;
14     const headBase2Y = headY - (scaledLineWidth / 2) * perpY;
15
16     this.context.beginPath();
17     this.context.moveTo(headBase1X, headBase1Y);
18     this.context.lineTo(headBase2X, headBase2Y);
19     this.context.lineTo(tailX, tailY);
20     this.context.closePath();
21     this.context.fillStyle = tailGradient;
22     this.context.fill();
23 }
```



# Designing the comet

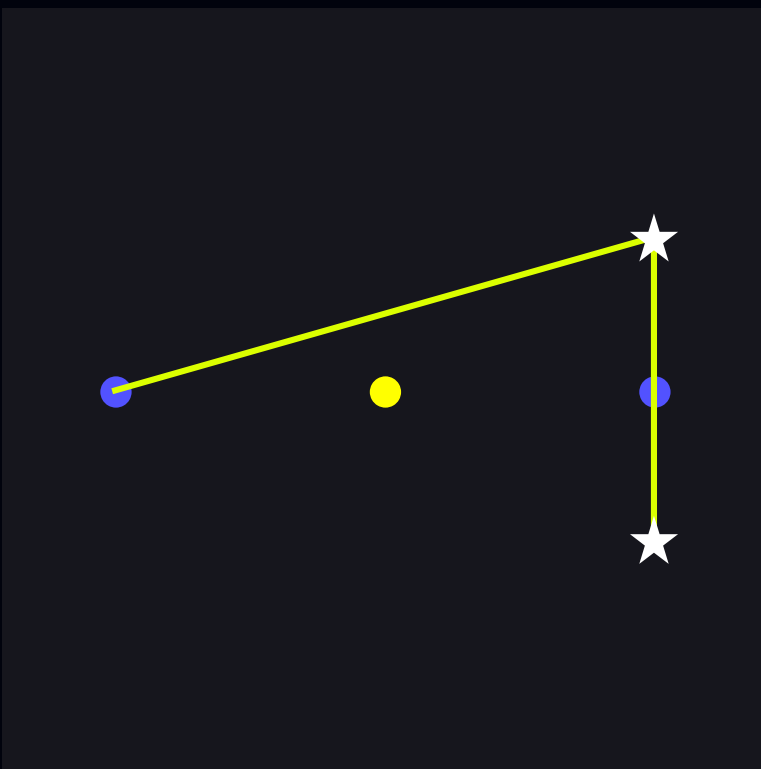
## Draw the tail



```
1  drawComet() {  
2    ...  
3  
4    // 1. Draw the tail  
5    const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);  
6    tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");  
7    tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");  
8  
9    const perpX = -Math.sin(this.comet.angle);  
10   const perpY = Math.cos(this.comet.angle);  
11   const headBase1X = headX + (scaledLineWidth / 2) * perpX;  
12   const headBase1Y = headY + (scaledLineWidth / 2) * perpY;  
13   const headBase2X = headX - (scaledLineWidth / 2) * perpX;  
14   const headBase2Y = headY - (scaledLineWidth / 2) * perpY;  
15  
16   this.context.beginPath();  
17   this.context.moveTo(headBase1X, headBase1Y);  
18   this.context.lineTo(headBase2X, headBase2Y);  
19   this.context.lineTo(tailX, tailY);  
20   this.context.closePath();  
21   this.context.fillStyle = tailGradient;  
22   this.context.fill();  
23 }
```

# Designing the comet

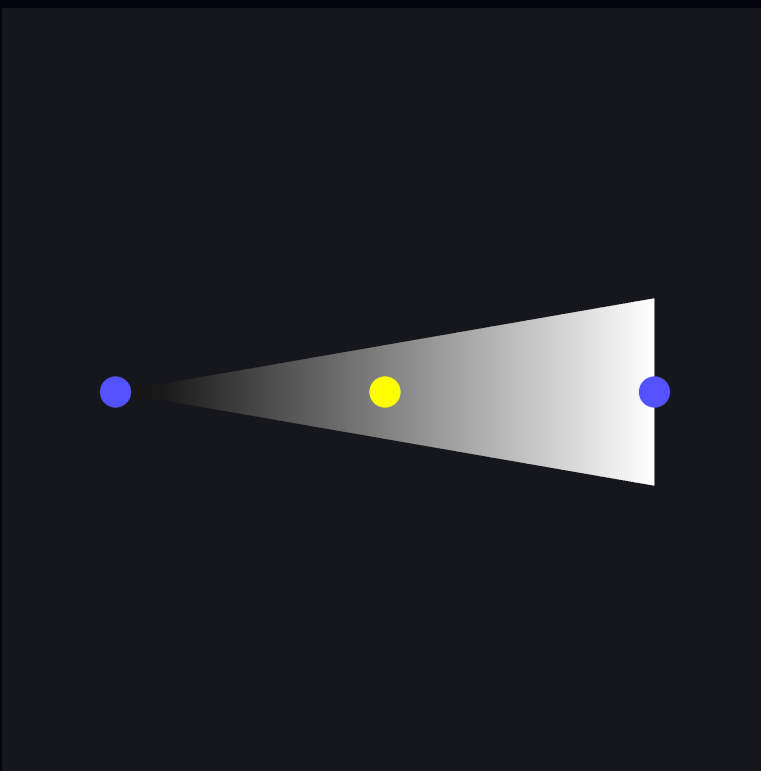
## Draw the tail



```
1  drawComet() {  
2    ...  
3  
4    // 1. Draw the tail  
5    const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);  
6    tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");  
7    tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");  
8  
9    const perpX = -Math.sin(this.comet.angle);  
10   const perpY = Math.cos(this.comet.angle);  
11   const headBase1X = headX + (scaledLineWidth / 2) * perpX;  
12   const headBase1Y = headY + (scaledLineWidth / 2) * perpY;  
13   const headBase2X = headX - (scaledLineWidth / 2) * perpX;  
14   const headBase2Y = headY - (scaledLineWidth / 2) * perpY;  
15  
16   this.context.beginPath();  
17   this.context.moveTo(headBase1X, headBase1Y);  
18   this.context.lineTo(headBase2X, headBase2Y);  
19   this.context.lineTo(tailX, tailY);  
20   this.context.closePath();  
21   this.context.fillStyle = tailGradient;  
22   this.context.fill();  
23 }
```

# Designing the comet

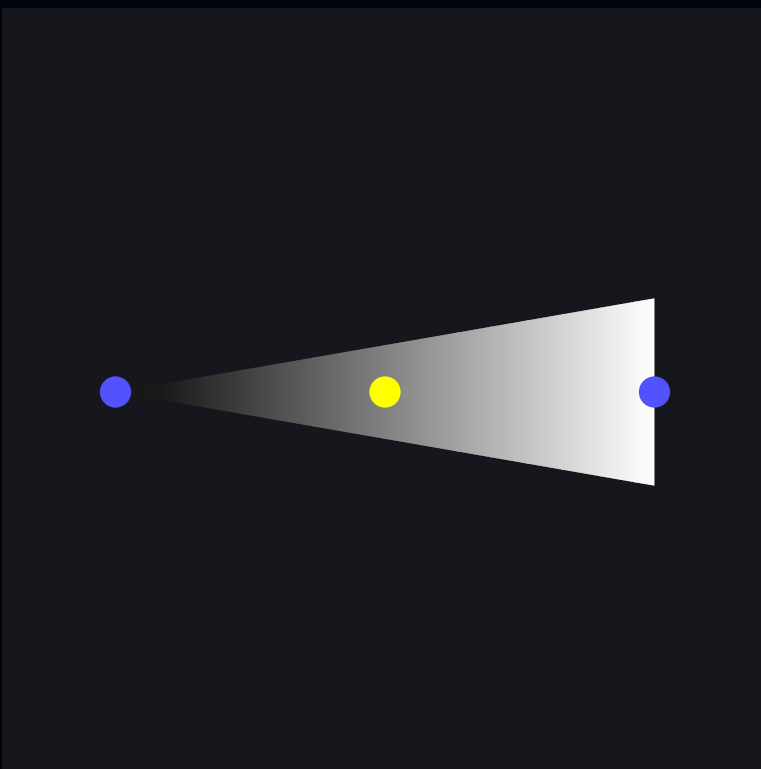
## Draw the tail



```
1  drawComet() {  
2      ...  
3  
4      // 1. Draw the tail  
5      const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);  
6      tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");  
7      tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");  
8  
9      const perpX = -Math.sin(this.comet.angle);  
10     const perpY = Math.cos(this.comet.angle);  
11     const headBase1X = headX + (scaledLineWidth / 2) * perpX;  
12     const headBase1Y = headY + (scaledLineWidth / 2) * perpY;  
13     const headBase2X = headX - (scaledLineWidth / 2) * perpX;  
14     const headBase2Y = headY - (scaledLineWidth / 2) * perpY;  
15  
16     this.context.beginPath();  
17     this.context.moveTo(headBase1X, headBase1Y);  
18     this.context.lineTo(headBase2X, headBase2Y);  
19     this.context.lineTo(tailX, tailY);  
20     this.context.closePath();  
21     this.context.fillStyle = tailGradient;  
22     this.context.fill();  
23 }
```

# Designing the comet

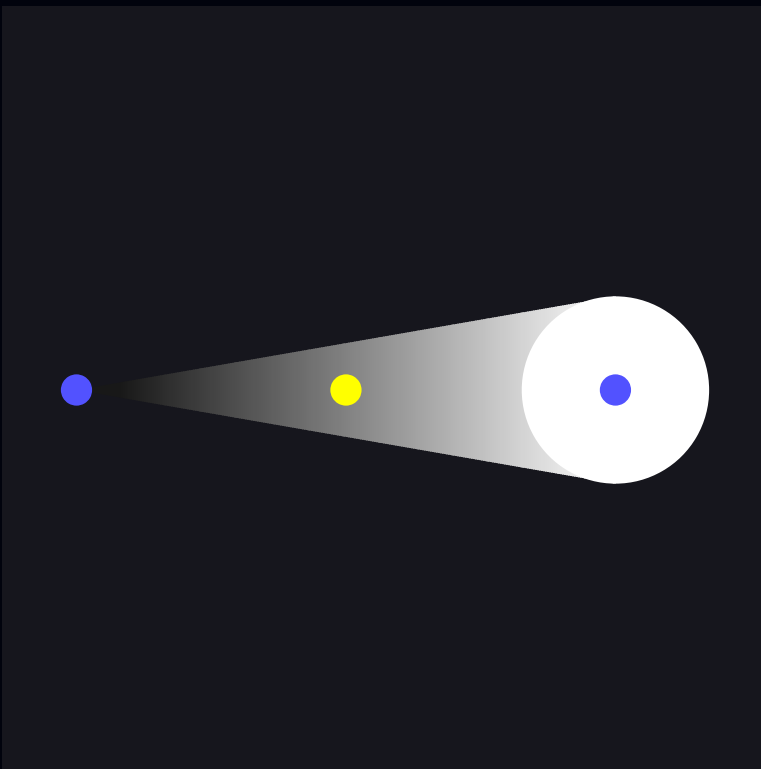
## Draw the tail



```
1  drawComet() {  
2      ...  
3  
4      // 1. Draw the tail  
5      const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);  
6      tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");  
7      tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");  
8  
9      const perpX = -Math.sin(this.comet.angle);  
10     const perpY = Math.cos(this.comet.angle);  
11     const headBase1X = headX + (scaledLineWidth / 2) * perpX;  
12     const headBase1Y = headY + (scaledLineWidth / 2) * perpY;  
13     const headBase2X = headX - (scaledLineWidth / 2) * perpX;  
14     const headBase2Y = headY - (scaledLineWidth / 2) * perpY;  
15  
16     this.context.beginPath();  
17     this.context.moveTo(headBase1X, headBase1Y);  
18     this.context.lineTo(headBase2X, headBase2Y);  
19     this.context.lineTo(tailX, tailY);  
20     this.context.closePath();  
21     this.context.fillStyle = tailGradient;  
22     this.context.fill();  
23 }
```

# Designing the comet

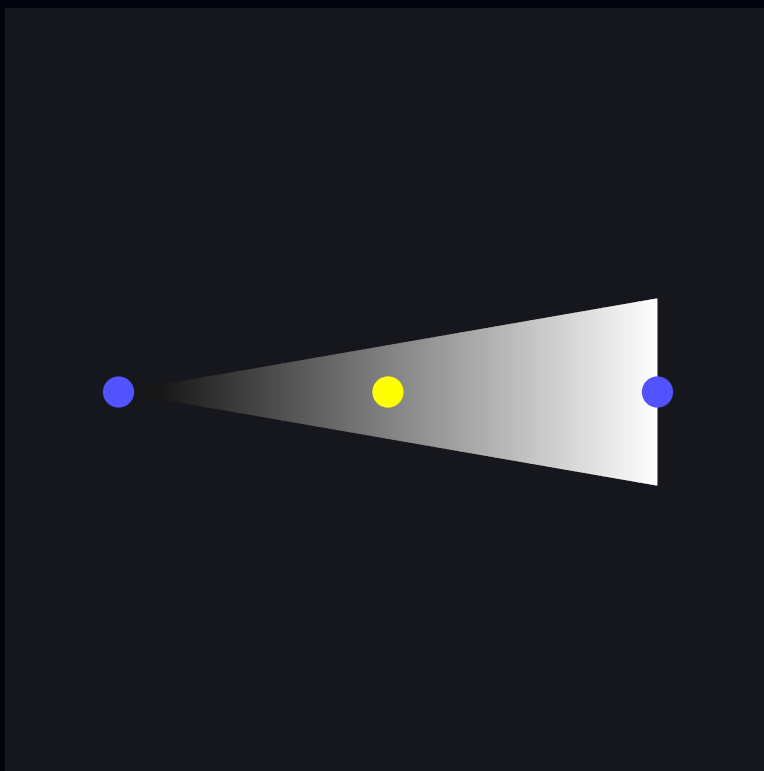
Draw the head



```
1  drawComet() {  
2      ...  
3  
4      // 2. Draw a solid head  
5      this.context.beginPath();  
6      this.context.arc(headX, headY, scaledLineWidth / 2, 0, 2 * Math.PI);  
7  
8      this.context.fillStyle = "white";  
9      this.context.fill();  
10 }
```

# Designing the comet

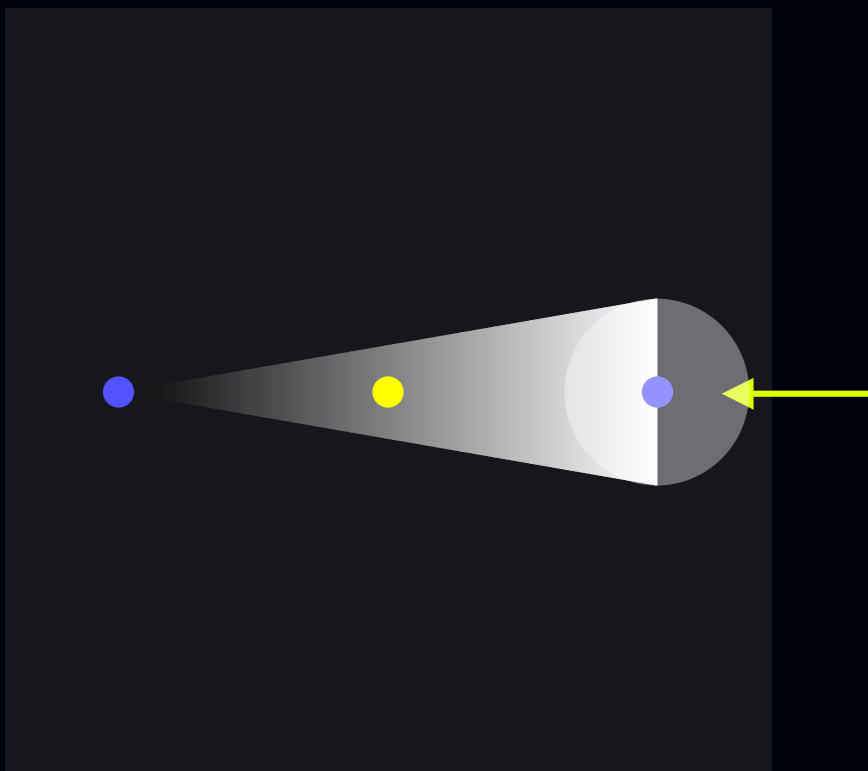
Draw the head



```
1  drawComet() {  
2      ...  
3  
4      // 2. Draw a solid head  
5      this.context.beginPath();  
6      this.context.arc(headX, headY, scaledLineWidth / 2, 0, 2 * Math.PI);  
7  
8      this.context.fillStyle = "white";  
9      this.context.fill();  
10 }
```

# Designing the comet

Draw the head



```
1  drawComet() {  
2      ...  
3  
4      // 2. Draw a solid head  
5      this.context.beginPath();  
6      this.context.arc(headX, headY, scaledLineWidth / 2, 0, 2 * Math.PI);  
7  
8      this.context.fillStyle = "white";  
9      this.context.fill();  
10 }
```

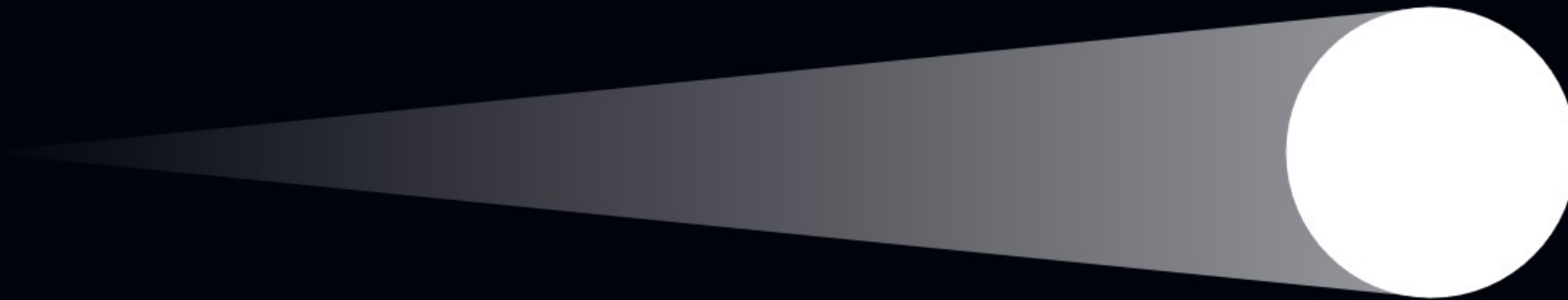
# Designing the comet

Final result

**Width = 10**



**Width = 50**





# Meteor shower

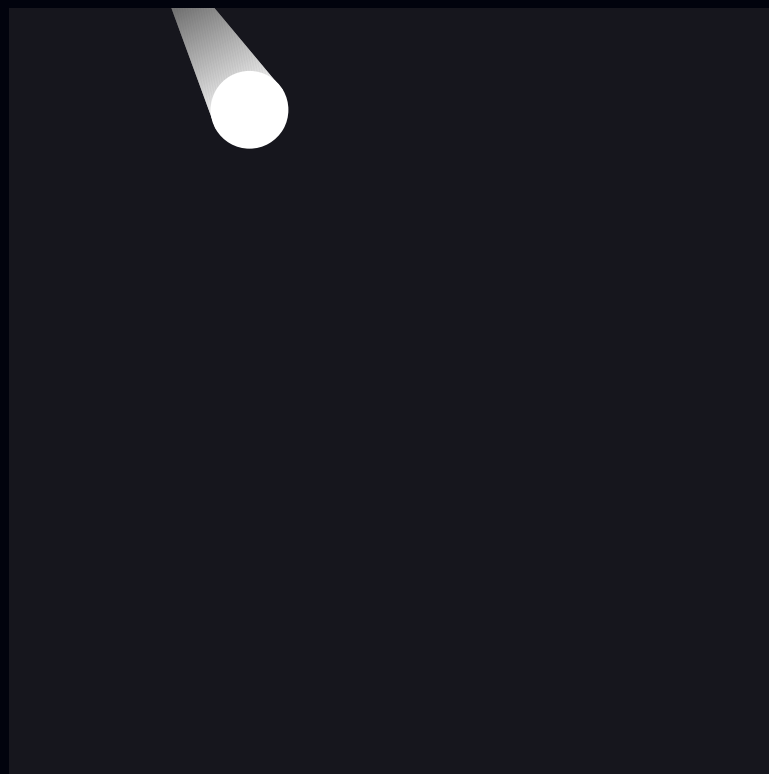
Code example



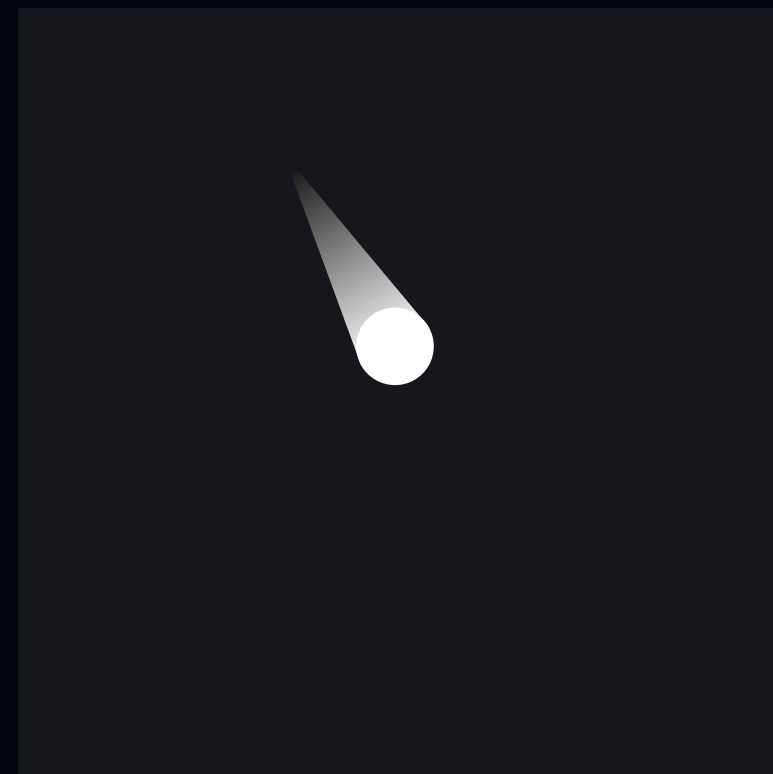
# Moving the comet

After spawning a comet, we will move it **in steps**.

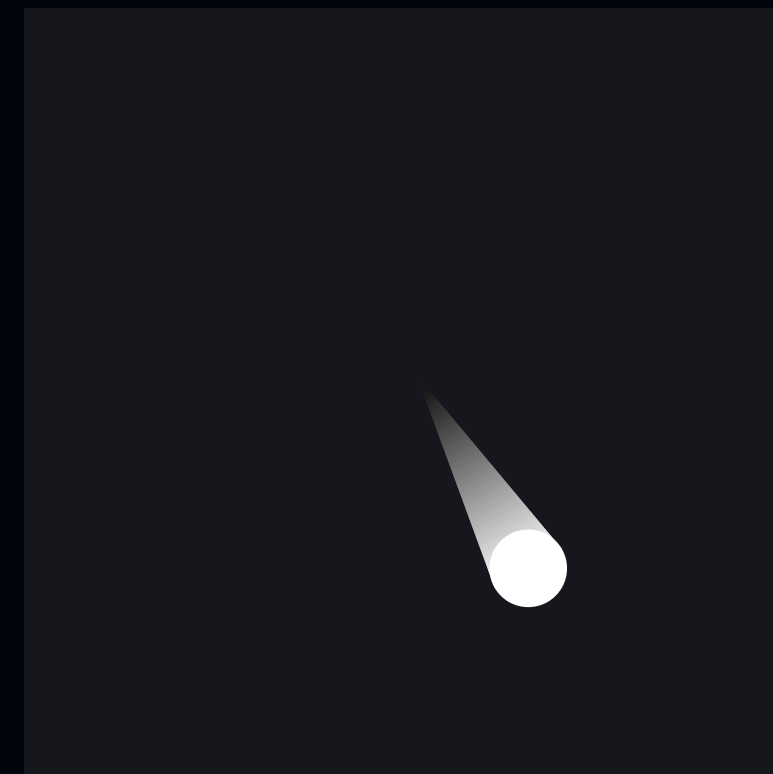
Example:



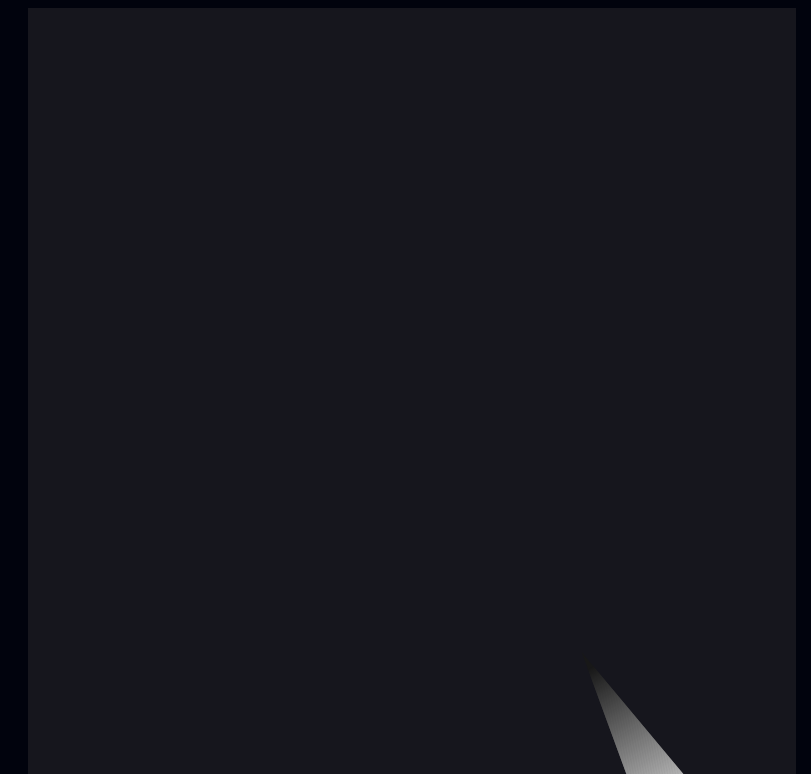
**Step 0**



**Step 1**



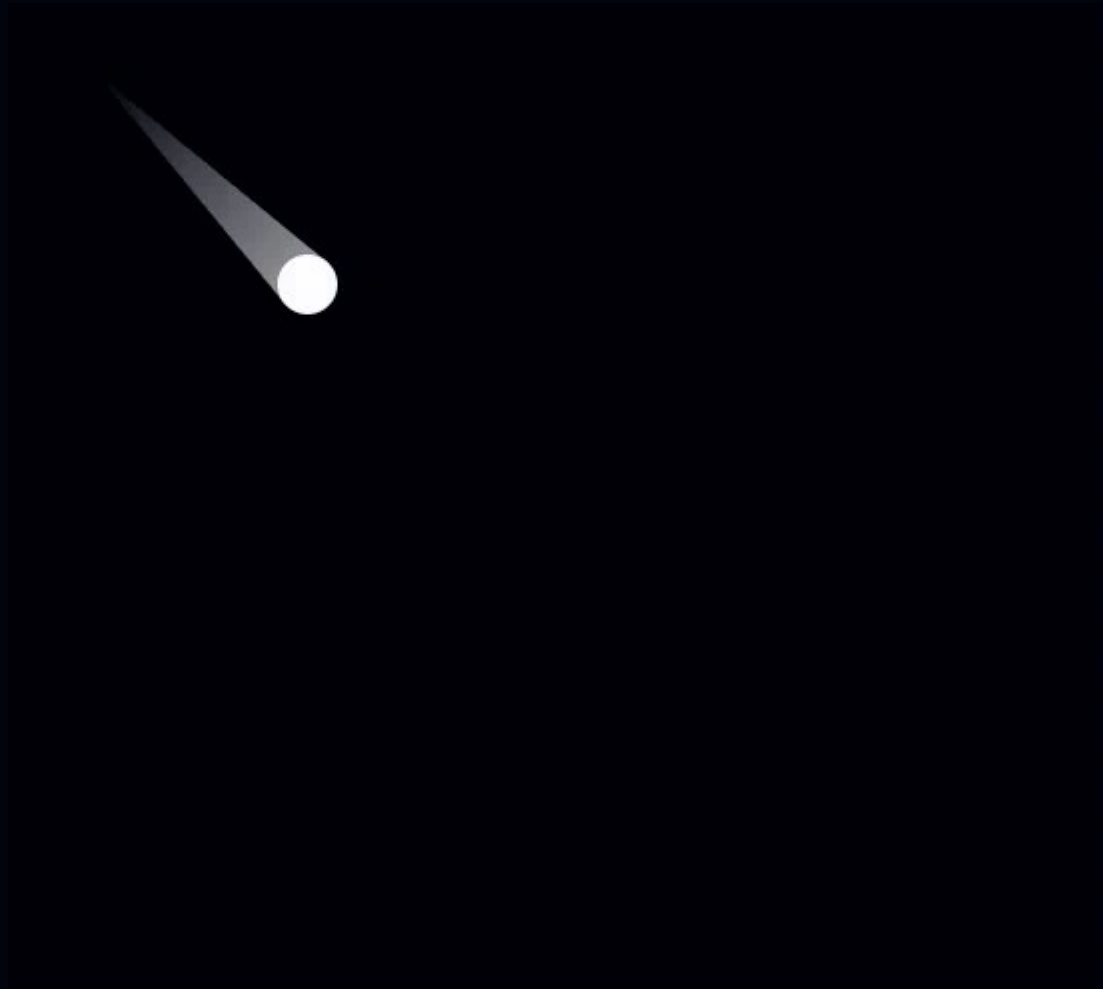
**Step 2**



**Step 3**

# Moving the comet

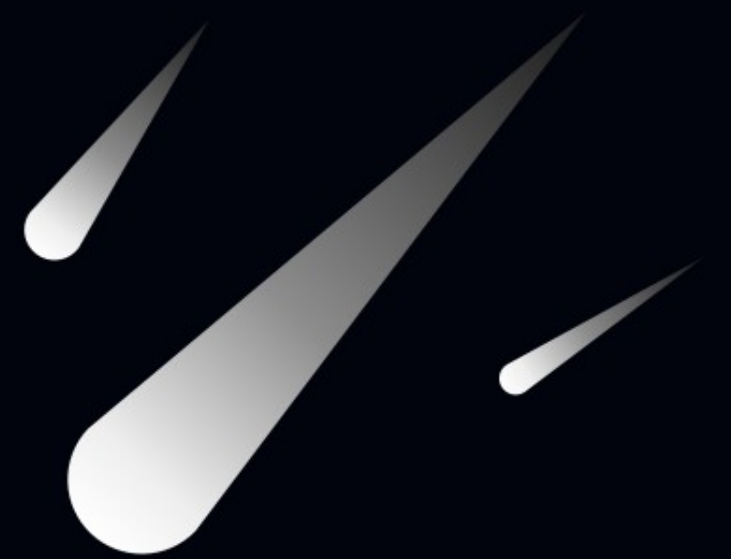
Code example:



```
1  drawComet() { ...
2  },
3  handleKeyPress(event) {
4      if (event.key === "Enter") {
5          this.stepAnimation();
6      }
7  },
8  stepAnimation() {
9      this.comet.positionX += this.comet.stepSizeX;
10     this.comet.positionY += this.comet.stepSizeY;
11
12     this.drawComet();
13 },
14
15 mounted() {
16     this.setupCanvas();
17     this.drawComet();
18
19     window.addEventListener("keydown", this.handleKeyPress);
20 },
21
```

# Meteor shower

Code example



# Meteor shower

To simulate a **meteor shower**, we will spawn multiple comets.

The meteor shower source is called **radiant**.



```
1  spawnMeteor() {  
2      const spawnZoneWidth = this.width * this.SPAWN_ZONE_WIDTH_SCALE;  
3      const spawnZoneOffset = (this.width - spawnZoneWidth) / 2;  
4      ...  
5  },  
6
```

# Meteor shower

To simulate a **meteor shower**, we will spawn multiple comets.

The meteor shower source is called **radiant**.



```
1  spawnMeteor() {  
2    const spawnZoneWidth = this.width * this.SPAWN_ZONE_WIDTH_SCALE;  
3    const spawnZoneOffset = (this.width - spawnZoneWidth) / 2;  
4    ...  
5  },  
6  
7  mounted() {  
8    this.radiant = { positionX: this.width / 2, positionY:  
9    this.RADIANT_POSITION_Y };  
10  },
```

# Meteor shower

To simulate a **meteor shower**, we will spawn multiple comets.

The meteor shower source is called **radiant**.

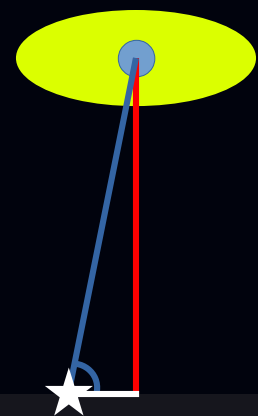


```
1  spawnMeteor() {  
2      const spawnZoneWidth = this.width * this.SPAWN_ZONE_WIDTH_SCALE;  
3      const spawnZoneOffset = (this.width - spawnZoneWidth) / 2;  
4  
5      const startX = Math.random() * spawnZoneWidth + spawnZoneOffset;  
6      const startY = 0;  
7      const angle = Math.atan2(startY - this.radiant.positionY, startX -  
8          this.radiant.positionX);  
9      ...  
10 }
```

# Meteor shower

To simulate a **meteor shower**, we will spawn multiple comets.

The meteor shower source is called **radiant**.



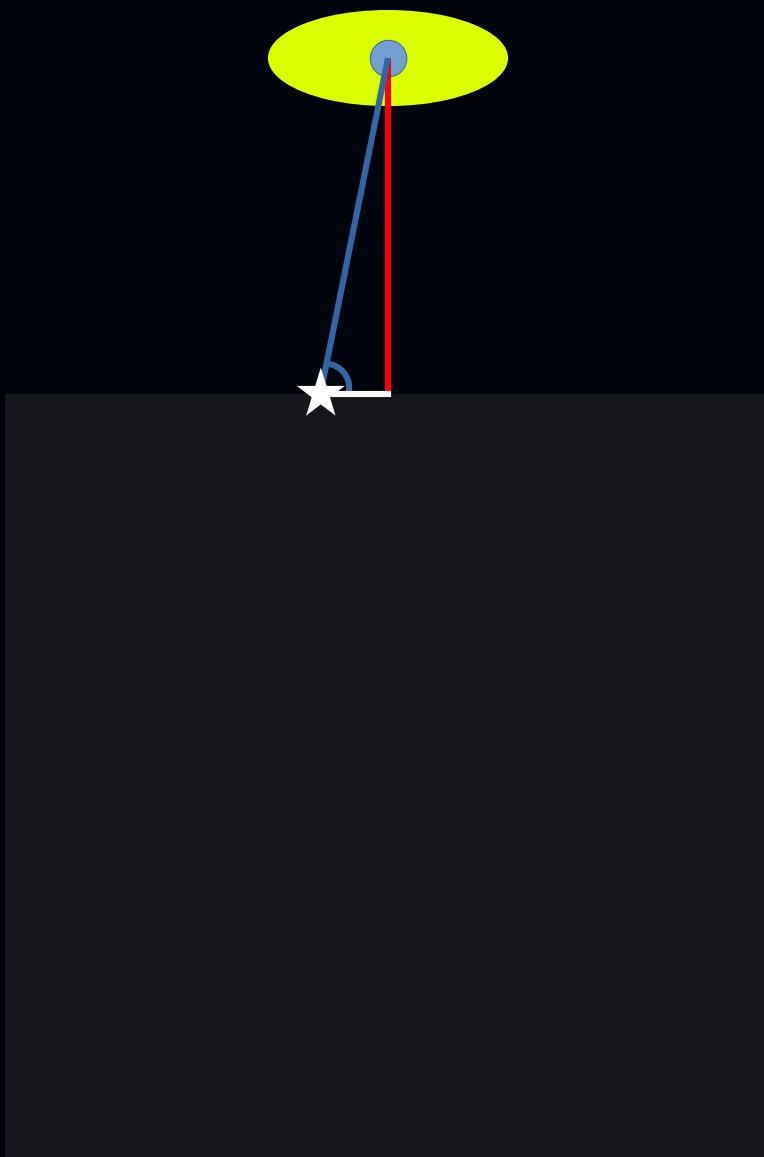
```
1  spawnMeteor() {  
2      const spawnZoneWidth = this.width * this.SPAWN_ZONE_WIDTH_SCALE;  
3      const spawnZoneOffset = (this.width - spawnZoneWidth) / 2;  
4  
5      const startX = Math.random() * spawnZoneWidth + spawnZoneOffset;  
6      const startY = 0;  
7      const angle = Math.atan2(startY - this.radiant.positionY, startX -  
8          this.radiant.positionX);  
9      ...  
10 }
```



# Meteor shower

To simulate a **meteor shower**, we will spawn multiple comets.

The meteor shower source is called **radiant**.



```
1  spawnMeteor() {  
2    const spawnZoneWidth = this.width * this.SPAWN_ZONE_WIDTH_SCALE;  
3    const spawnZoneOffset = (this.width - spawnZoneWidth) / 2;  
4    ...  
5    this.meteors.push({  
6      positionX: startX,  
7      positionY: startY,  
8      stepSizeX: Math.cos(angle) * speed,  
9      stepSizeY: Math.sin(angle) * speed,  
10     length,  
11     lineWidth,  
12     opacity,  
13   });  
14   ...  
15 }
```

# Meteor shower

To simulate a **meteor shower**, we will spawn multiple comets.

The meteor shower source is called **radiant**.

```
1  drawMeteors() {  
2    this.context.clearRect(0, 0, this.width, this.height);  
3    for (let i = this.meteors.length - 1; i >= 0; i--) {  
4      const meteor = this.meteors[i];  
5      ...  
6    },  
7    this.animationFrameId = requestAnimationFrame(this.drawMeteors);  
8  },
```

# Meteor shower

Code example

