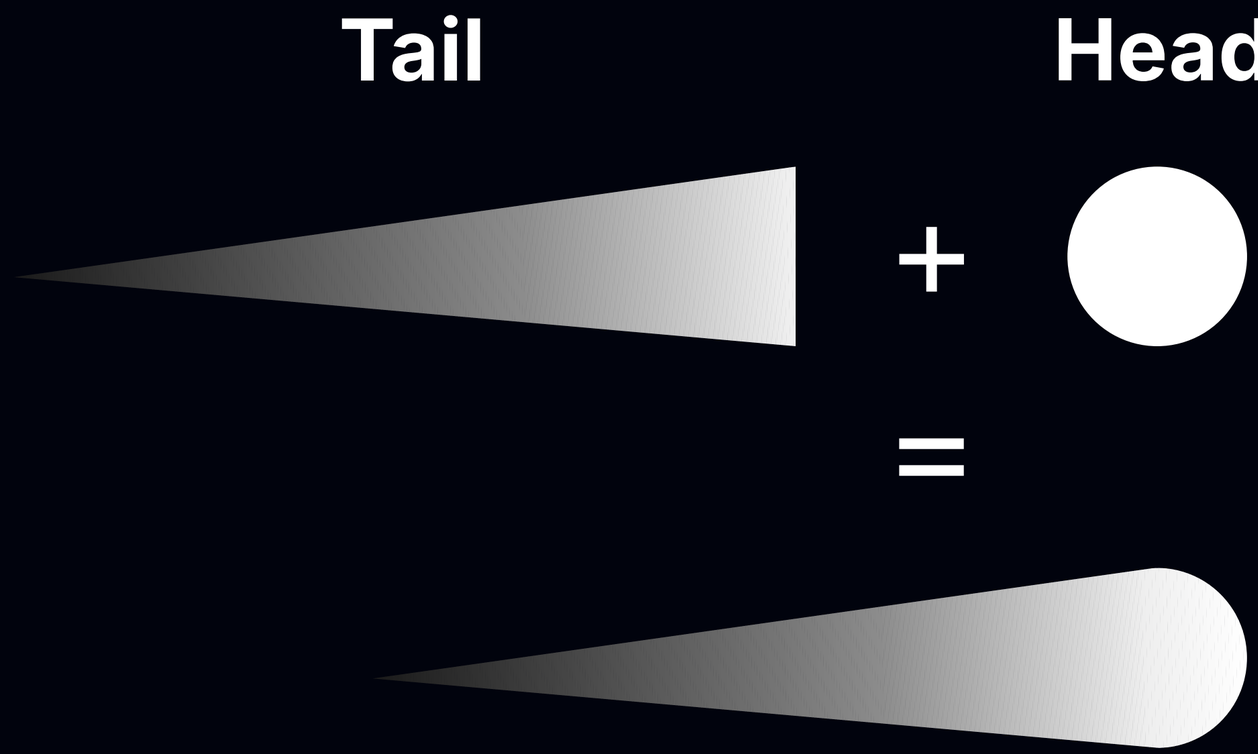
Meteor shower

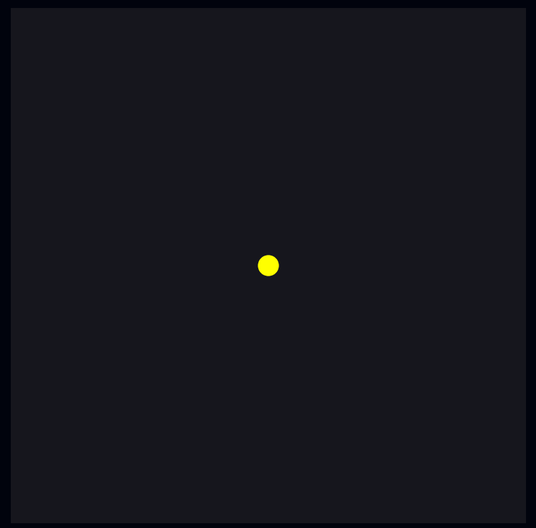
Designing the comet



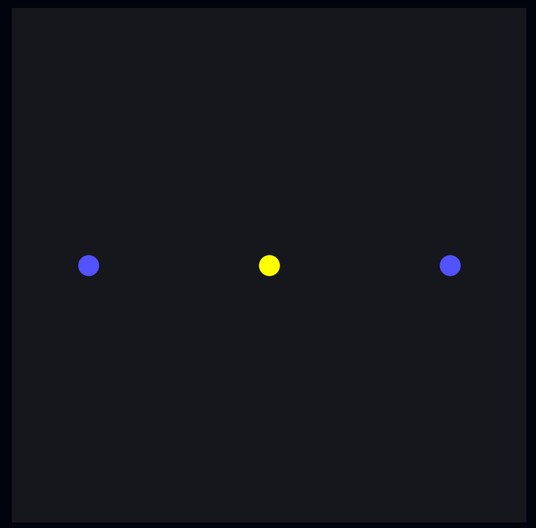
Designing the comet

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

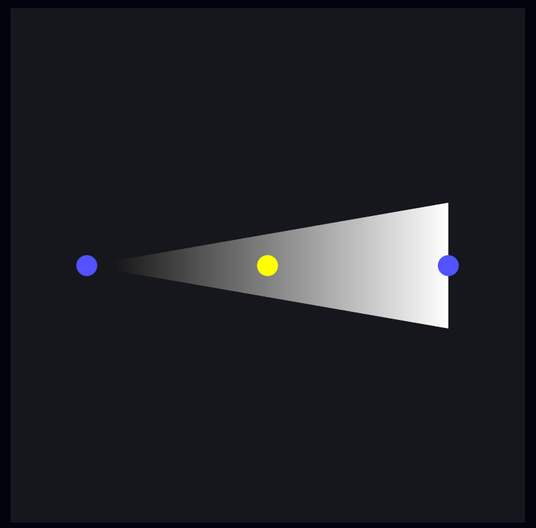
The steps to follow:



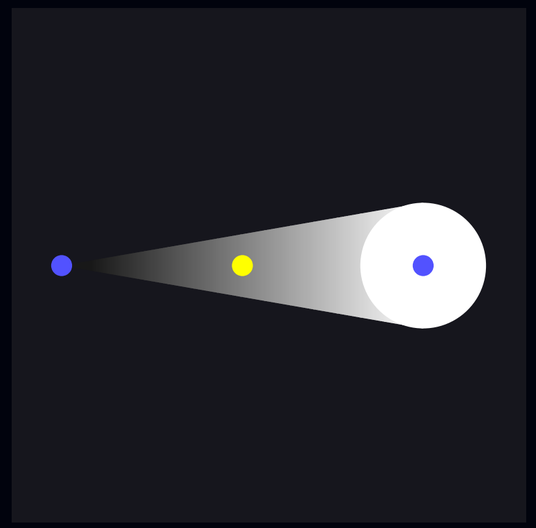
Find the center



Locate the head and tail



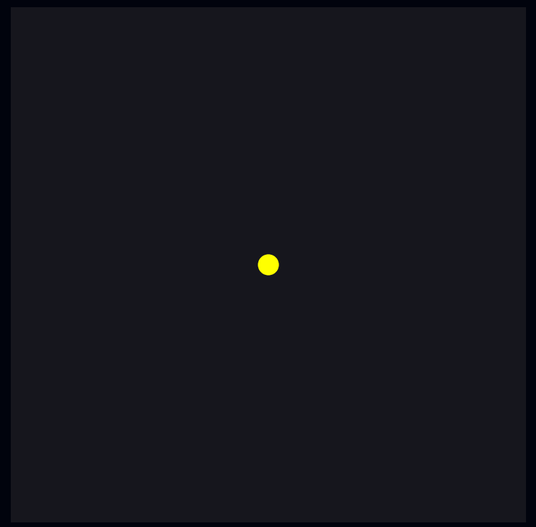
Draw the tail



Add the head

Designing the comet

Find the center



```
drawComet() {
  this.context.clearRect(0, 0, this.width, this.height);

  const centerX = this.width / 2;
  const centerY = this.height / 2;

  // Draw the center point
  this.context.beginPath();
  this.context.arc(centerX, centerY, 8, 0, 2 * Math.PI);
  this.context.fillStyle = "yellow";
  this.context.fill();
},
```

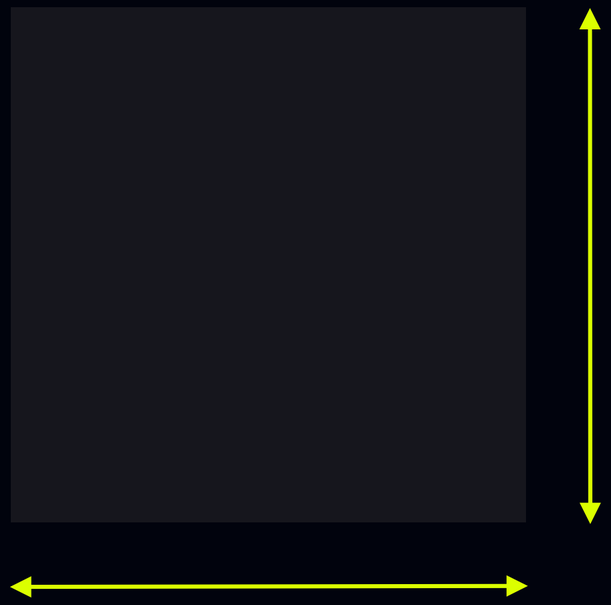
Designing the comet

Find the center

```
drawComet() {
  this.context.clearRect(0, 0, this.width, this.height);

  const centerX = this.width / 2;
  const centerY = this.height / 2;

  // Draw the center point
  this.context.beginPath();
  this.context.arc(centerX, centerY, 8, 0, 2 * Math.PI);
  this.context.fillStyle = "yellow";
  this.context.fill();
},
```

Designing the comet

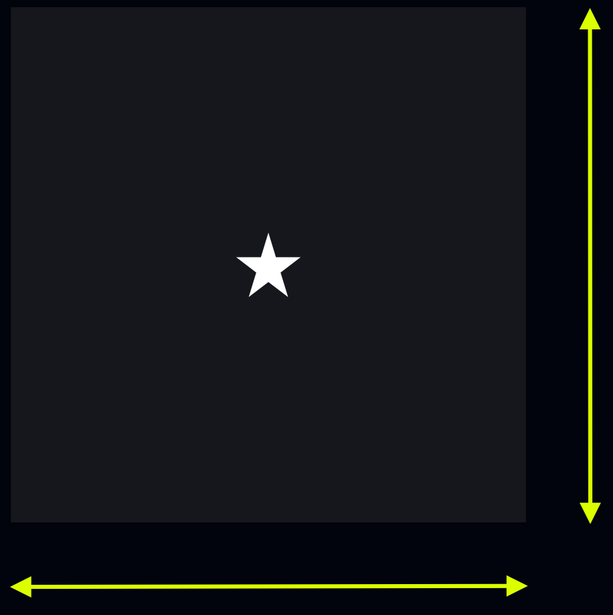
Find the center



```
drawComet() {
  this.context.clearRect(0, 0, this.width, this.height);

  const centerX = this.width / 2;
  const centerY = this.height / 2;

  // Draw the center point
  this.context.beginPath();
  this.context.arc(centerX, centerY, 8, 0, 2 * Math.PI);
  this.context.fillStyle = "yellow";
  this.context.fill();
},
```

Designing the comet

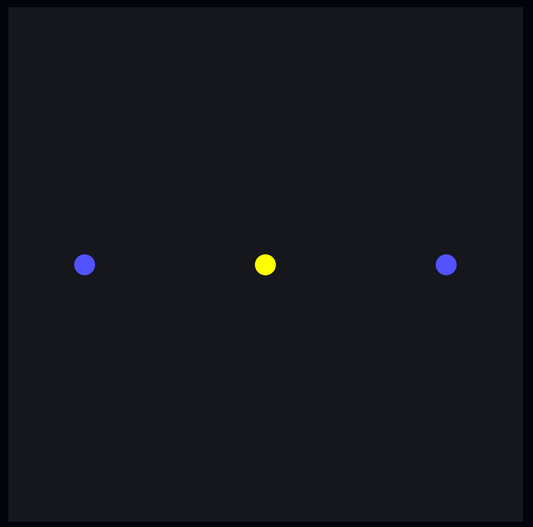
Find the center



```
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

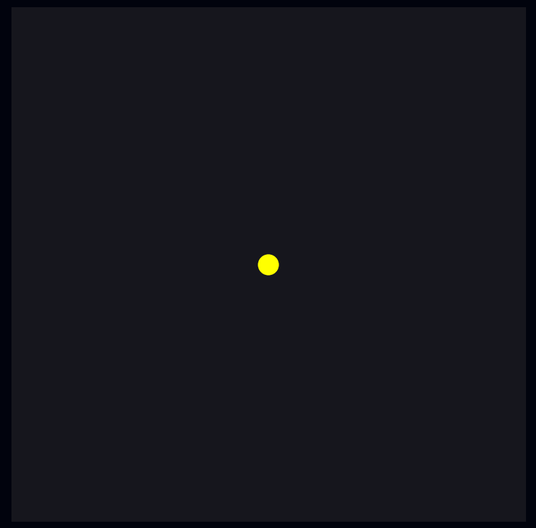
Locate the head and tail



```
drawComet() {
  ...

  const scaledLength = this.comet.length * this.scale;
  const scaledLineWidth = this.comet.lineWidth * this.scale;

  const headX = centerX + (scaledLength / 2) * Math.cos(this.comet.angle);
  const headY = centerY + (scaledLength / 2) * Math.sin(this.comet.angle);

  // Draw the point at the head
  this.context.beginPath();
  this.context.arc(headX, headY, 8, 0, 2 * Math.PI);
  this.context.fillStyle = "blue";
  this.context.fill();

  const tailX = centerX - (scaledLength / 2) * Math.cos(this.comet.angle);
  const tailY = centerY - (scaledLength / 2) * Math.sin(this.comet.angle);

  // Draw the point at the tail end
  this.context.beginPath();
  this.context.arc(tailX, tailY, 8, 0, 2 * Math.PI);
  this.context.fillStyle = "blue";
  this.context.fill();
},
```

Designing the comet

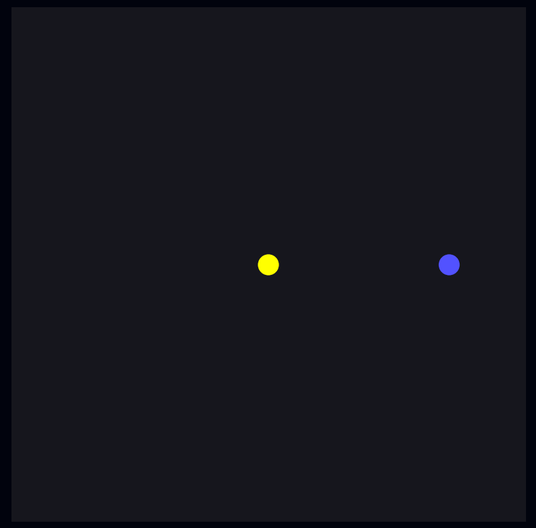
Locate the head and tail



```
drawComet() {
  ...

  const scaledLength = this.comet.length * this.scale;
  const scaledLineWidth = this.comet.lineWidth * this.scale;

  const headX = centerX + (scaledLength / 2)
  const headY = centerY + (scaledLength / 2)

  // Draw the point at the head
  this.context.beginPath();
  this.context.arc(headX, headY, 8, 0, 2 * Math.PI);
  this.context.fillStyle = "blue";
  this.context.fill();

  const tailX = centerX - (scaledLength / 2)
  const tailY = centerY - (scaledLength / 2)

  // Draw the point at the tail end
  this.context.beginPath();
  this.context.arc(tailX, tailY, 8, 0, 2 * Math.PI);
  this.context.fillStyle = "blue";
  this.context.fill();
},
```

Designing the comet

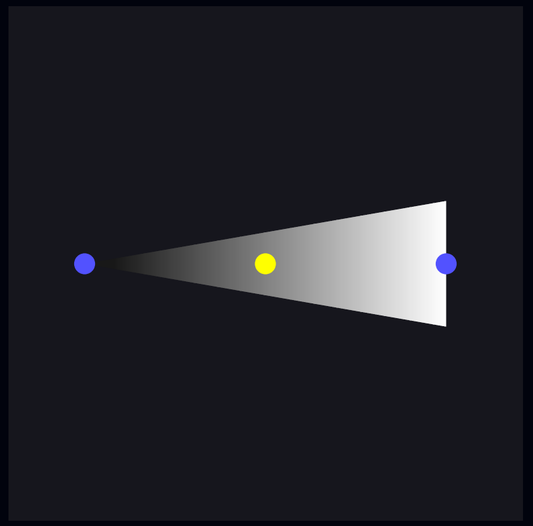
Locate the head and tail



```
drawComet() {
  ...

  const scaledLength = this.comet.length * this.scale;
  const scaledLineWidth = this.comet.lineWidth * this.scale;

  const headX = centerX + (scaledLength / 2)
  const headY = centerY + (scaledLength / 2)

  // Draw the point at the head
  this.context.beginPath();
  this.context.arc(headX, headY, 8, 0, 2 * Math.PI);
  this.context.fillStyle = "blue";
  this.context.fill();

  const tailX = centerX - (scaledLength / 2)
  const tailY = centerY - (scaledLength / 2)

  // Draw the point at the tail end
  this.context.beginPath();
  this.context.arc(tailX, tailY, 8, 0, 2 * Math.PI);
  this.context.fillStyle = "blue";
  this.context.fill();
},
```

Designing the comet

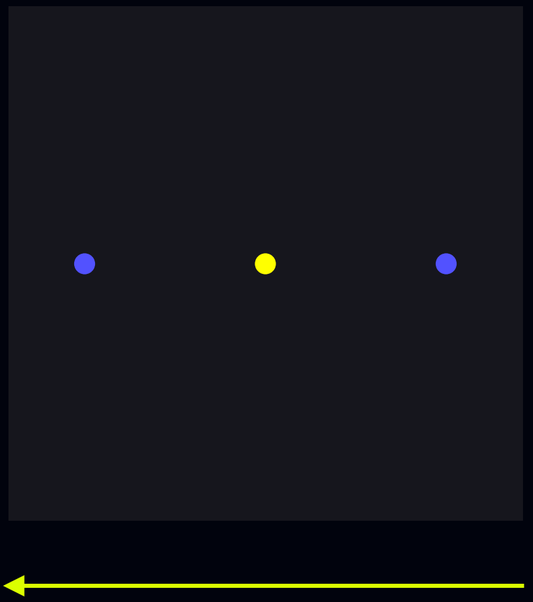
Draw the tail



```
drawComet() {
  ...

  // 1. Draw the tail
  const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);
  tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");
  tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");

  const perpX = -Math.sin(this.comet.angle);
  const perpY = Math.cos(this.comet.angle);
  const headBase1X = headX + (scaledLineWidth / 2) * perpX;
  const headBase1Y = headY + (scaledLineWidth / 2) * perpY;
  const headBase2X = headX - (scaledLineWidth / 2) * perpX;
  const headBase2Y = headY - (scaledLineWidth / 2) * perpY;

  this.context.beginPath();
  this.context.moveTo(headBase1X, headBase1Y);
  this.context.lineTo(headBase2X, headBase2Y);
  this.context.lineTo(tailX, tailY);
  this.context.closePath();
  this.context.fillStyle = tailGradient;
  this.context.fill();
},
```

Designing the comet

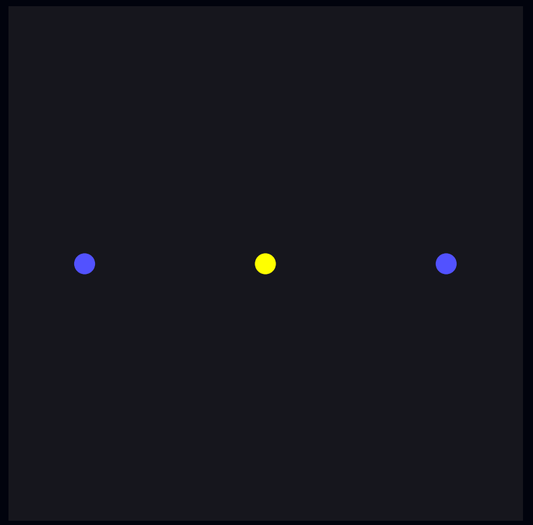
Draw the tail



```
drawComet() {
  ...

  // 1. Draw the tail
  const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);
  tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");
  tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");

  const perpX = -Math.sin(this.comet.angle);
  const perpY = Math.cos(this.comet.angle);
  const headBase1X = headX + (scaledLineWidth / 2) * perpX;
  const headBase1Y = headY + (scaledLineWidth / 2) * perpY;
  const headBase2X = headX - (scaledLineWidth / 2) * perpX;
  const headBase2Y = headY - (scaledLineWidth / 2) * perpY;

  this.context.beginPath();
  this.context.moveTo(headBase1X, headBase1Y);
  this.context.lineTo(headBase2X, headBase2Y);
  this.context.lineTo(tailX, tailY);
  this.context.closePath();
  this.context.fillStyle = tailGradient;
  this.context.fill();
},
```

Designing the comet

Draw the tail

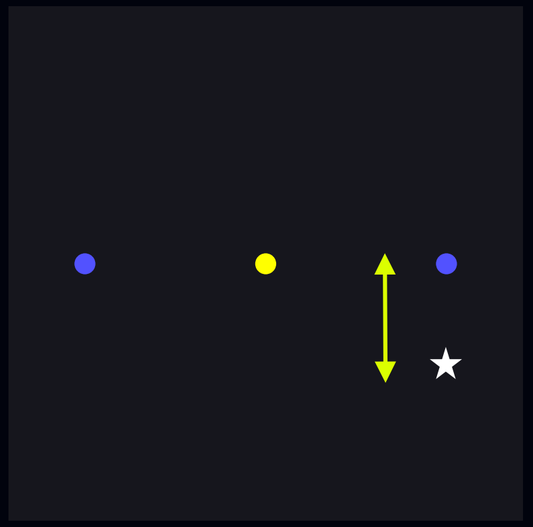


perpX = Sin(0) = 0
perpY = Cos(0) = 1

```
drawComet() {
  ...

  // 1. Draw the tail
  const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);
  tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");
  tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");

  const perpX = -Math.sin(this.comet.angle);
  const perpY = Math.cos(this.comet.angle);
  const headBase1X = headX + (scaledLineWidth / 2) * perpX;
  const headBase1Y = headY + (scaledLineWidth / 2) * perpY;
  const headBase2X = headX - (scaledLineWidth / 2) * perpX;
  const headBase2Y = headY - (scaledLineWidth / 2) * perpY;

  this.context.beginPath();
  this.context.moveTo(headBase1X, headBase1Y);
  this.context.lineTo(headBase2X, headBase2Y);
  this.context.lineTo(tailX, tailY);
  this.context.closePath();
  this.context.fillStyle = tailGradient;
  this.context.fill();
},
```

Designing the comet

Draw the tail



perpX = Sin(0) = 0
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

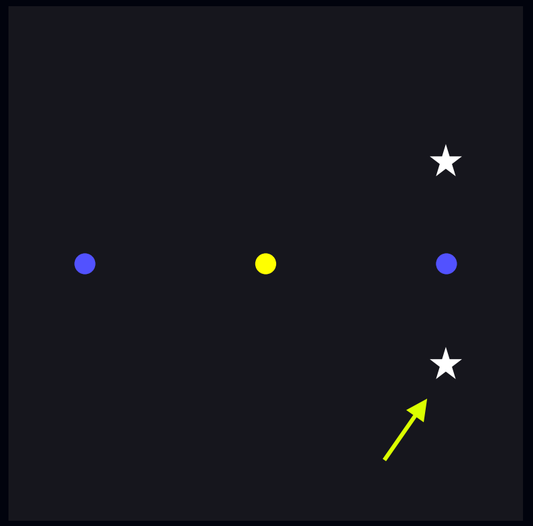
Draw the tail

perpX = Sin(0) = 0
perpY = Cos(0) = 1

```
drawComet() {
  ...

  // 1. Draw the tail
  const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);
  tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");
  tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");

  const perpX = -Math.sin(this.comet.angle);
  const perpY = Math.cos(this.comet.angle);
  const headBase1X = headX + (scaledLineWidth / 2) * perpX;
  const headBase1Y = headY + (scaledLineWidth / 2) * perpY;
  const headBase2X = headX - (scaledLineWidth / 2) * perpX;
  const headBase2Y = headY - (scaledLineWidth / 2) * perpY;

  this.context.beginPath();
  this.context.moveTo(headBase1X, headBase1Y);
  this.context.lineTo(headBase2X, headBase2Y);
  this.context.lineTo(tailX, tailY);
  this.context.closePath();
  this.context.fillStyle = tailGradient;
  this.context.fill();
},
```

Designing the comet

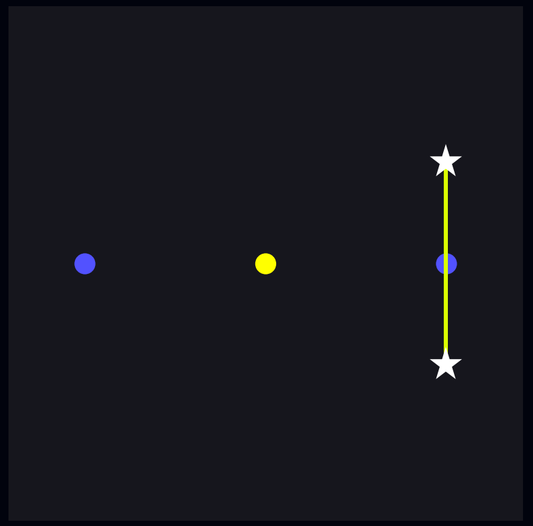
Draw the tail



```
drawComet() {
  ...

  // 1. Draw the tail
  const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);
  tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");
  tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");

  const perpX = -Math.sin(this.comet.angle);
  const perpY = Math.cos(this.comet.angle);
  const headBase1X = headX + (scaledLineWidth / 2) * perpX;
  const headBase1Y = headY + (scaledLineWidth / 2) * perpY;
  const headBase2X = headX - (scaledLineWidth / 2) * perpX;
  const headBase2Y = headY - (scaledLineWidth / 2) * perpY;

  this.context.beginPath();
  this.context.moveTo(headBase1X, headBase1Y);
  this.context.lineTo(headBase2X, headBase2Y);
  this.context.lineTo(tailX, tailY);
  this.context.closePath();
  this.context.fillStyle = tailGradient;
  this.context.fill();
},
```

Designing the comet

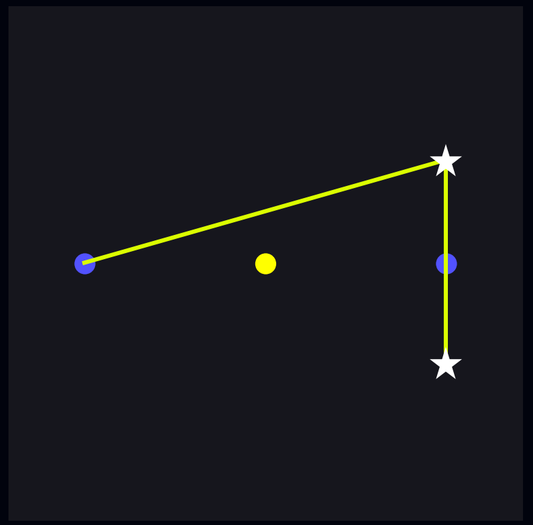
Draw the tail



```
drawComet() {
  ...

  // 1. Draw the tail
  const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);
  tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");
  tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");

  const perpX = -Math.sin(this.comet.angle);
  const perpY = Math.cos(this.comet.angle);
  const headBase1X = headX + (scaledLineWidth / 2) * perpX;
  const headBase1Y = headY + (scaledLineWidth / 2) * perpY;
  const headBase2X = headX - (scaledLineWidth / 2) * perpX;
  const headBase2Y = headY - (scaledLineWidth / 2) * perpY;

  this.context.beginPath();
  this.context.moveTo(headBase1X, headBase1Y);
  this.context.lineTo(headBase2X, headBase2Y);
  this.context.lineTo(tailX, tailY);
  this.context.closePath();
  this.context.fillStyle = tailGradient;
  this.context.fill();
},
```

Designing the comet

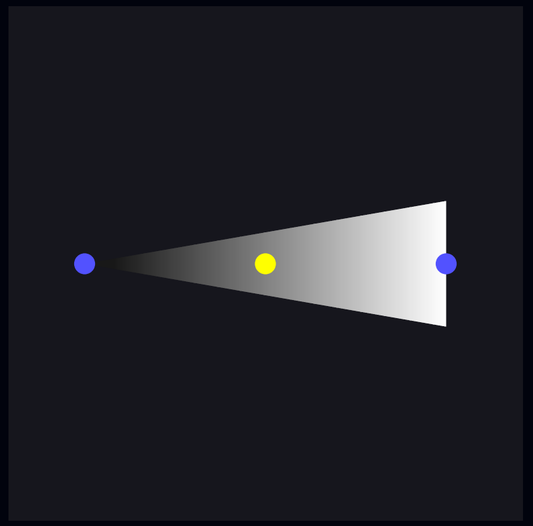
Draw the tail



```
drawComet() {
  ...

  // 1. Draw the tail
  const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);
  tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");
  tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");

  const perpX = -Math.sin(this.comet.angle);
  const perpY = Math.cos(this.comet.angle);
  const headBase1X = headX + (scaledLineWidth / 2) * perpX;
  const headBase1Y = headY + (scaledLineWidth / 2) * perpY;
  const headBase2X = headX - (scaledLineWidth / 2) * perpX;
  const headBase2Y = headY - (scaledLineWidth / 2) * perpY;

  this.context.beginPath();
  this.context.moveTo(headBase1X, headBase1Y);
  this.context.lineTo(headBase2X, headBase2Y);
  this.context.lineTo(tailX, tailY);
  this.context.closePath();
  this.context.fillStyle = tailGradient;
  this.context.fill();
},
```

Designing the comet

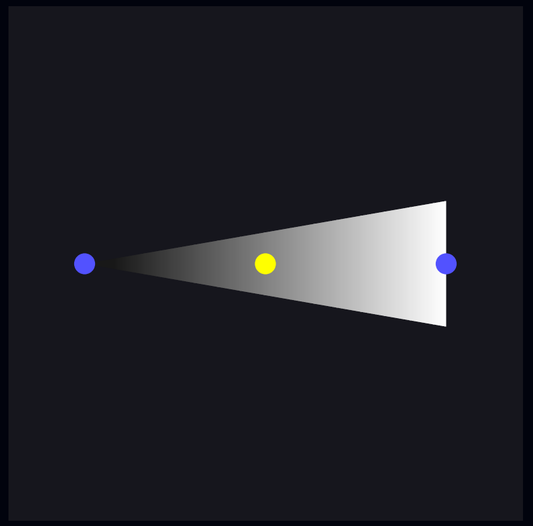
Draw the tail



```
drawComet() {
  ...

  // 1. Draw the tail
  const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);
  tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");
  tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");

  const perpX = -Math.sin(this.comet.angle);
  const perpY = Math.cos(this.comet.angle);
  const headBase1X = headX + (scaledLineWidth / 2) * perpX;
  const headBase1Y = headY + (scaledLineWidth / 2) * perpY;
  const headBase2X = headX - (scaledLineWidth / 2) * perpX;
  const headBase2Y = headY - (scaledLineWidth / 2) * perpY;

  this.context.beginPath();
  this.context.moveTo(headBase1X, headBase1Y);
  this.context.lineTo(headBase2X, headBase2Y);
  this.context.lineTo(tailX, tailY);
  this.context.closePath();
  this.context.fillStyle = tailGradient;
  this.context.fill();
},
```

Designing the comet

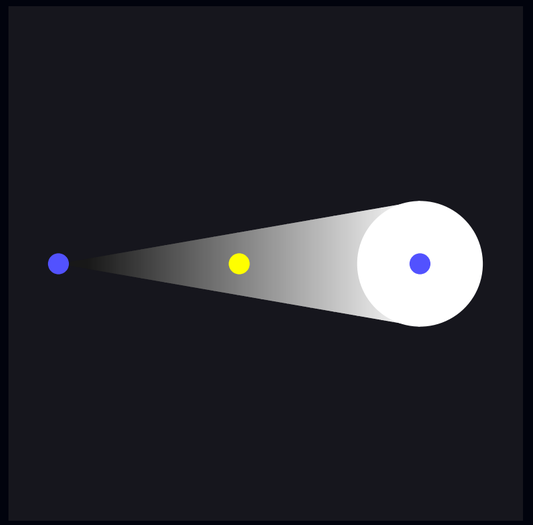
Draw the tail



```
drawComet() {
  ...

  // 1. Draw the tail
  const tailGradient = this.context.createLinearGradient(headX, headY, tailX, tailY);
  tailGradient.addColorStop(0, "rgba(255, 255, 255, 0.6)");
  tailGradient.addColorStop(1, "rgba(255, 255, 255, 0)");

  const perpX = -Math.sin(this.comet.angle);
  const perpY = Math.cos(this.comet.angle);
  const headBase1X = headX + (scaledLineWidth / 2) * perpX;
  const headBase1Y = headY + (scaledLineWidth / 2) * perpY;
  const headBase2X = headX - (scaledLineWidth / 2) * perpX;
  const headBase2Y = headY - (scaledLineWidth / 2) * perpY;

  this.context.beginPath();
  this.context.moveTo(headBase1X, headBase1Y);
  this.context.lineTo(headBase2X, headBase2Y);
  this.context.lineTo(tailX, tailY);
  this.context.closePath();
  this.context.fillStyle = tailGradient;
  this.context.fill();
},
```

Designing the comet

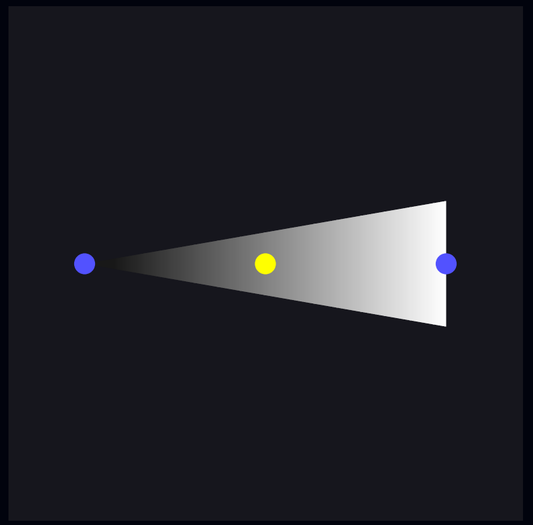
Draw the head



```
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

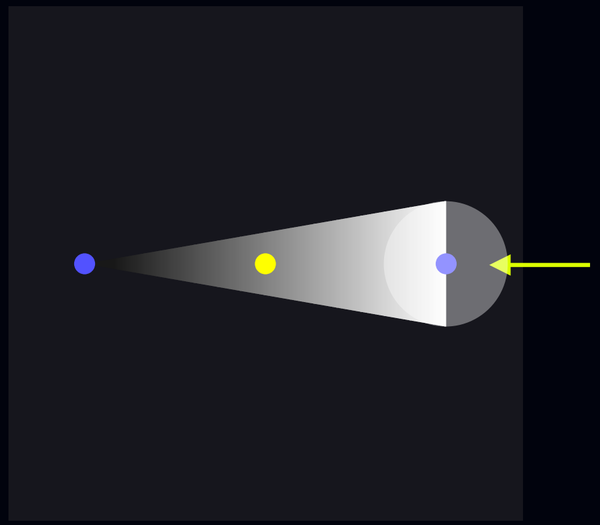
Draw the head



```
drawComet() {
  ...

  // 2. Draw a solid head
  this.context.beginPath();
  this.context.arc(headX, headY, scaledLineWidth / 2, 0, 2 * Math.PI);

  this.context.fillStyle = "white";
  this.context.fill();
},
```

Designing the comet

Draw the head



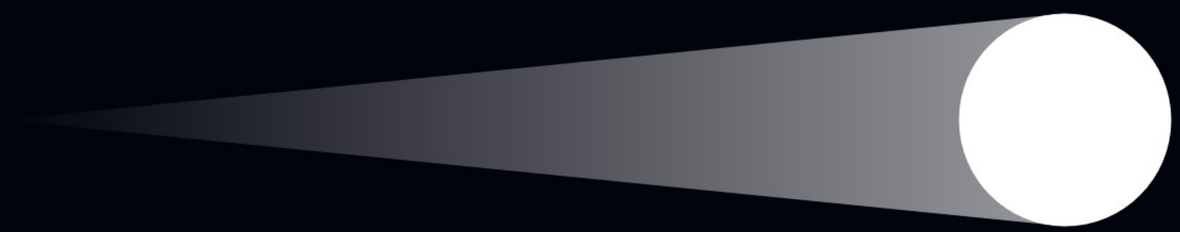
```
1   drawComet() {
2     ...
3
4     // 2. Draw a solid head
5     this.context.beginPath();
6     this.context.arc(headX, headY, scaledLineWidth / 2, 0, 2 * Math.PI);
7
8     this.context.fillStyle = "white";
9     this.context.fill();
10  },
```

Designing the comet

Final result

Width = 10

Width = 50



Meteor shower

Code example

Moving the comet

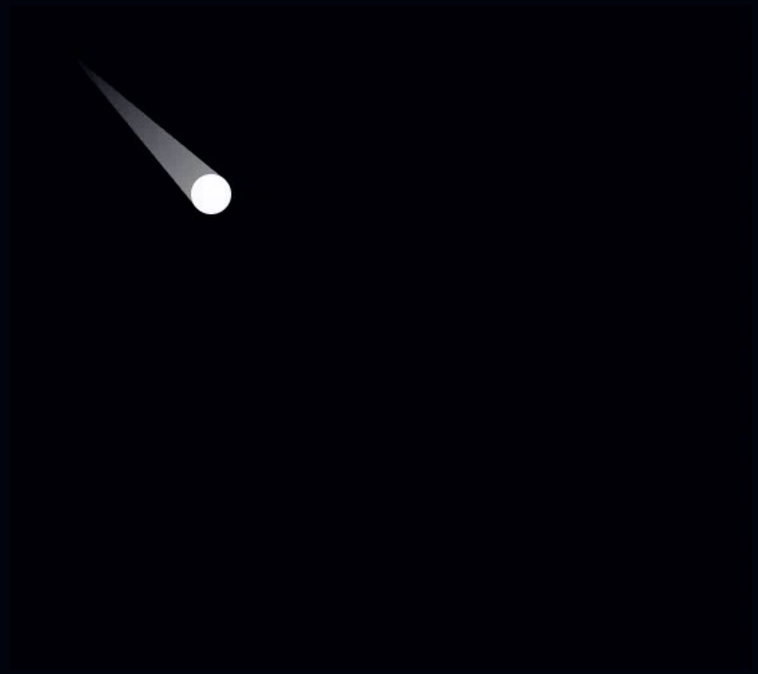
After spawning a comet, we will move it in steps.

Example:

Step 0 Step 1 Step 2 Step 3

Moving the comet

Code example:



```
drawComet() { ...
},
handleKeyPress(event) {
  if (event.key === "Enter") {
    this.stepAnimation();
  }
},
stepAnimation() {
  this.comet.positionX += this.comet.stepSizeX;
  this.comet.positionY += this.comet.stepSizeY;

  this.drawComet();
},

mounted() {
  this.setupCanvas();
  this.drawComet();

  window.addEventListener("keydown", this.handleKeyPress);
},

```

Meteor shower

Code example

Meteor shower

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

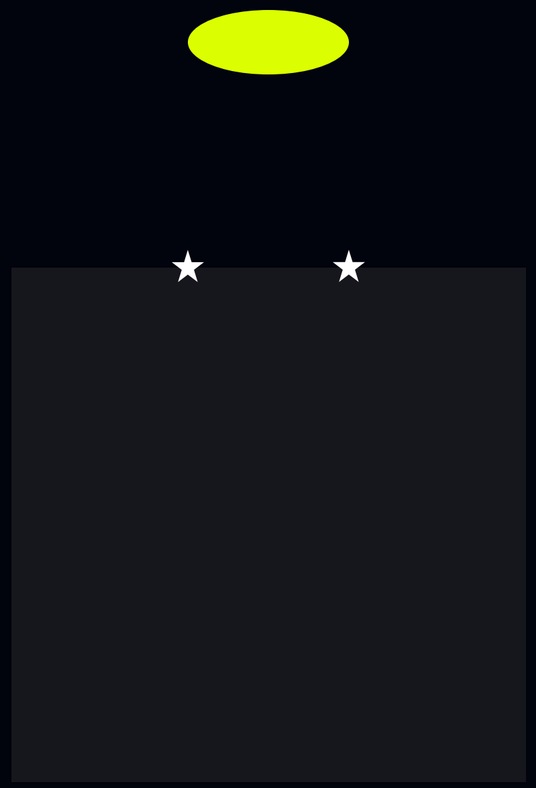
The meteor shower source is called radiant.

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

Meteor shower

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

The meteor shower source is called radiant.

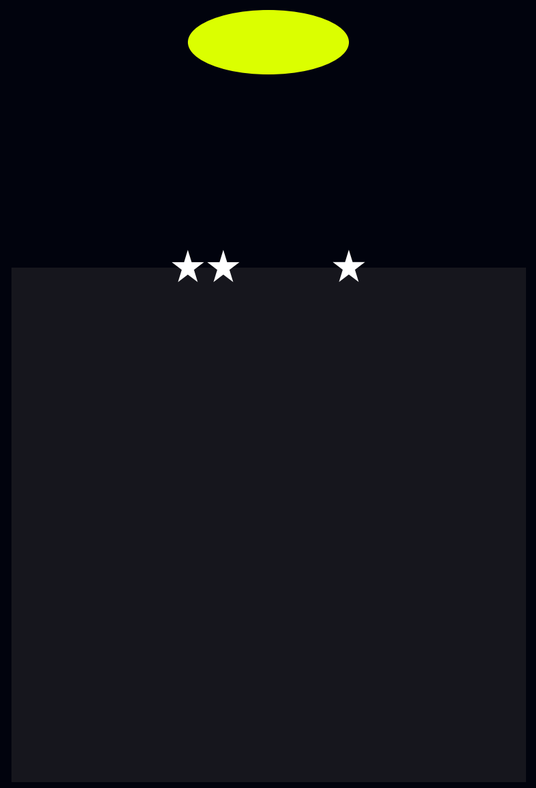


```
spawnMeteor() {
  const spawnZoneWidth = this.width * this.SPAWN_ZONE_WIDTH_SCALE;
  const spawnZoneOffset = (this.width - spawnZoneWidth) / 2;
  ...
},

mounted() {
  this.radiant = { positionX: this.width / 2, positionY:
this.RADIANT_POSITION_Y };
},
```

Meteor shower

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

The meteor shower source is called radiant.

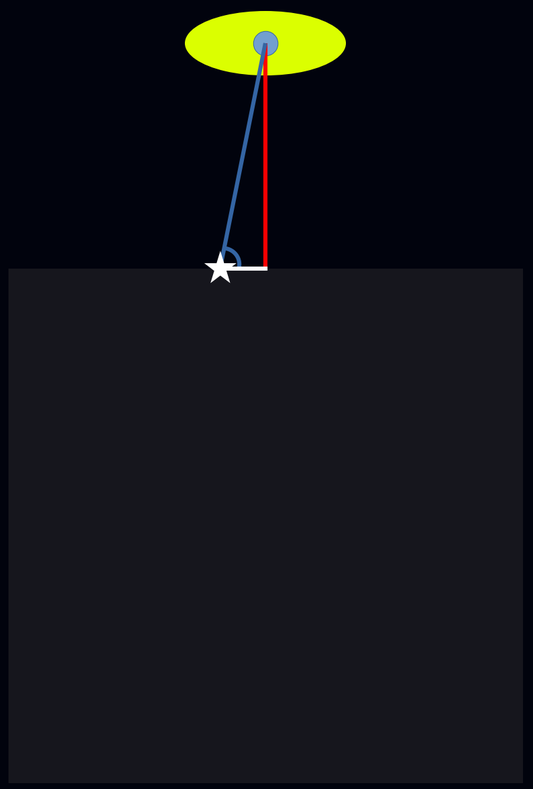


```
spawnMeteor() {
  const spawnZoneWidth = this.width * this.SPAWN_ZONE_WIDTH_SCALE;
  const spawnZoneOffset = (this.width - spawnZoneWidth) / 2;

  const startX = Math.random() * spawnZoneWidth + spawnZoneOffset;
  const startY = 0;
  const angle = Math.atan2(startY - this.radiant.positionY, startX -
        this.radiant.positionX);
  ...
},
```

Meteor shower

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

The meteor shower source is called radiant.

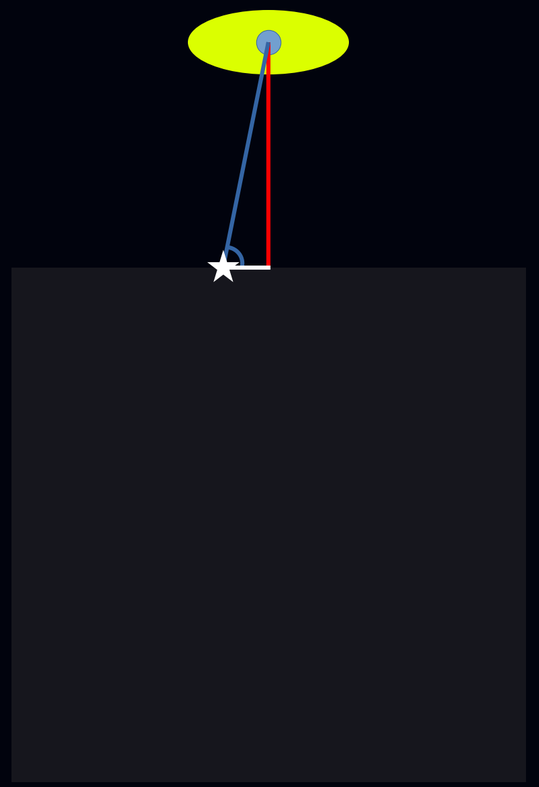


```
spawnMeteor() {
  const spawnZoneWidth = this.width * this.SPAWN_ZONE_WIDTH_SCALE;
  const spawnZoneOffset = (this.width - spawnZoneWidth) / 2;

  const startX = Math.random() * spawnZoneWidth + spawnZoneOffset;
  const startY = 0;
  const angle = Math.atan2(startY - this.radiant.positionY, startX -
        this.radiant.positionX);
  ...
},
```

Meteor shower

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

The meteor shower source is called radiant.



```
spawnMeteor() {
  const spawnZoneWidth = this.width * this.SPAWN_ZONE_WIDTH_SCALE;
  const spawnZoneOffset = (this.width - spawnZoneWidth) / 2;
  ...
  this.meteors.push({
    positionX: startX,
    positionY: startY,
    stepSizeX: Math.cos(angle) * speed,
    stepSizeY: Math.sin(angle) * speed,
    length,
    lineWidth,
    opacity,
  });
  ...
},
```

Meteor shower

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

The meteor shower source is called radiant.

```
drawMeteors() {
  this.context.clearRect(0, 0, this.width, this.height);
  for (let i = this.meteors.length - 1; i ⩽ 0; i--) {
    const meteor = this.meteors[i];
    ...
  },
  this.animationFrameId = requestAnimationFrame(this.drawMeteors);
},
```

Meteor shower

Code example