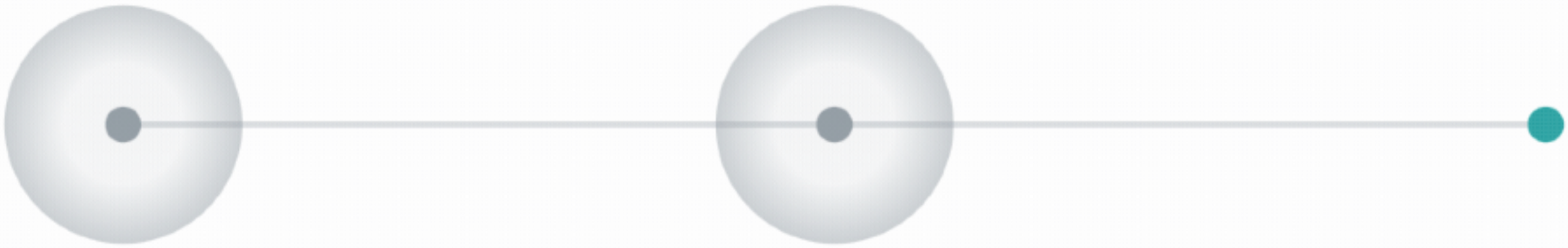
implementing the chart with

Customer



1
Active carts

1
Checking out

0
Purchased

Open custom settings

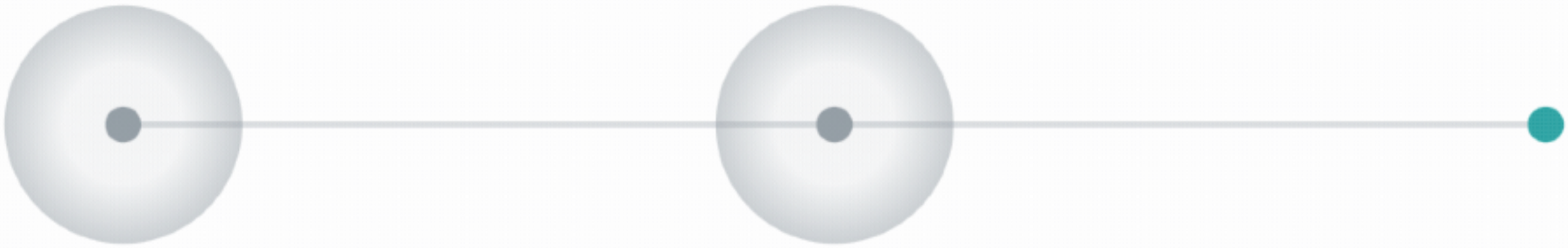
useEffect(() => {
if (!context) return;

```
    clearCanvas();
    drawAnimatedElements();
    drawStaticElements();
}, [bucketsRadius, ordersRunningDots]);
```

```
const drawAnimatedElements = () => {
  bucketsRadius.map((radius, index) => {
    const x = xScale(index);
    const y = maxRadius + PADDING;

    const gradient = context.createRadialGradient(x, y, 0, x, y, maxRadius);
    const fillColor = index === bucketsRadius.length - 1 ? green : gray;

    gradient.addColorStop(.5, rgba(fillColor, 0.1));
    gradient.addColorStop(1, rgba(fillColor, 0.4));

    drawCircle({
      context,
      x,
      y,
      radius,
      fillColor: gradient,
    });
  });

  ordersRunningDots.map(({ item, props, key }) => {

    drawCircle({
      context,
      x: props.cx.value,
      y: maxRadius + PADDING,
      radius: maxRadius / 10,
      fillColor: rgba(green, 0.5),
    });
  });
};
```

```
ordersRunningDots.map(({cx, fill, radius}, index) => {

  drawCircle({
    context,
    x: cx,
    y: maxRadius + PADDING,
    radius: radius,
    fillColor: fill,
  });
});
```

Customer



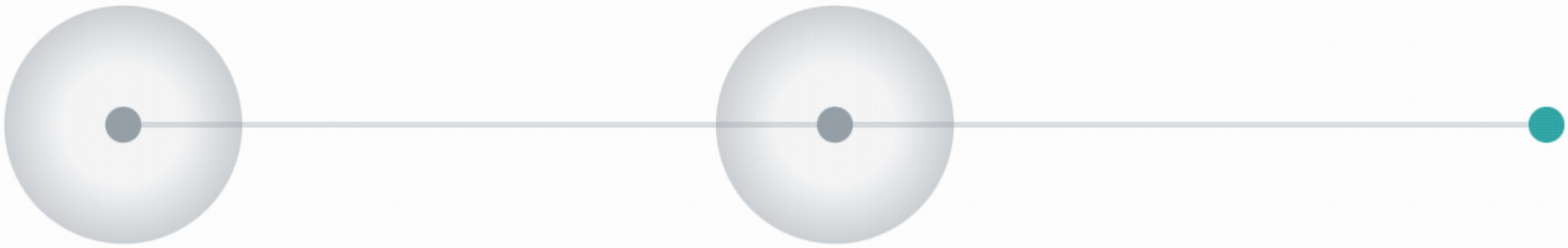
1
Active carts

1
Checking out

0
Purchased

Open custom settings

Customer



1
Active carts

1
Checking out

0
Purchased

Open custom settings

```
const drawAnimatedElements = () => {
  bucketsRadius.map((radius, index) => {
    const x = xScale(index);
    const y = maxRadius + PADDING;

    const gradient = context.createRadialGradient(x, y, 0, x, y, maxRadius);
    const fillColor = index === bucketsRadius.length - 1 ? green : gray;

    gradient.addColorStop(.5, rgba(fillColor, 0.1));
    gradient.addColorStop(1, rgba(fillColor, 0.4));

    drawCircle({
      context,
      x,
      y,
      radius,
      fillColor: gradient,
    });
  });

  ordersRunningDots.map(({ item, props, key }) => {

    drawCircle({
      context,
      x: props.cx.value,
      y: maxRadius + PADDING,
      radius: maxRadius / 10,
      fillColor: rgba(green, 0.5),
    });
  });
};
```

```
const drawAnimatedElements = () => {
  bucketsRadius.map((radius, index) => {
    const x = xScale(index);
    const y = maxRadius + PADDING;

    const gradient = context.createRadialGradient(x, y, 0, x, y, maxRadius);
    const fillColor = index === bucketsRadius.length - 1 ? green : gray;

    gradient.addColorStop(.5, rgba(fillColor, 0.1));
    gradient.addColorStop(1, rgba(fillColor, 0.4));

    drawCircle({
      context,
      x,
      y,
      radius,
      fillColor: gradient,
    });
  });

  ordersRunningDots.map(({ item, props, key }) => {

    drawCircle({
      context,
      x: props.cx.value,
      y: maxRadius + PADDING,
      radius: maxRadius / 10,
      fillColor: rgba(green, 0.5),
    });
  });
};
```

```
ordersRunningDots.map(({cx, fill, radius}, index) => {

  drawCircle({
    context,
    x: cx,
    y: maxRadius + PADDING,
    radius: radius,
    fillColor: fill,
  });
});
```