EECS 368
Programming Language Paradigms

David O. Johnson

Fall 2022

Reminders

- Assignment 3 due: 11:59 PM, Monday, October 3
- Assignment 4 due: 11:59 PM, Monday, October 17

Any Questions?

In-Class Problem Solution

- 12-(9-21) In-Class Problem Solution.pptx

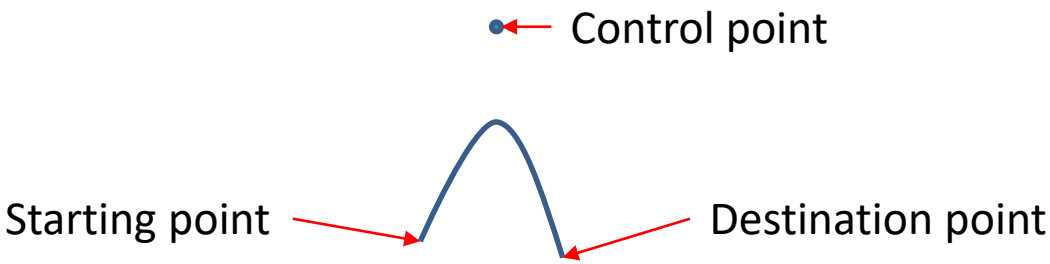
Any Questions?

Chapter 17 - Drawing on Canvas

- ~~Displaying Graphics~~
- ~~SVG~~
- ~~The canvas element~~
- ~~Lines and surfaces~~
- ~~Paths~~
- Curves
- Drawing a pie chart
- Text
- Images
- Transformation
- Storing and clearing transformations
- Choosing a graphics interface

Quadratic Curves

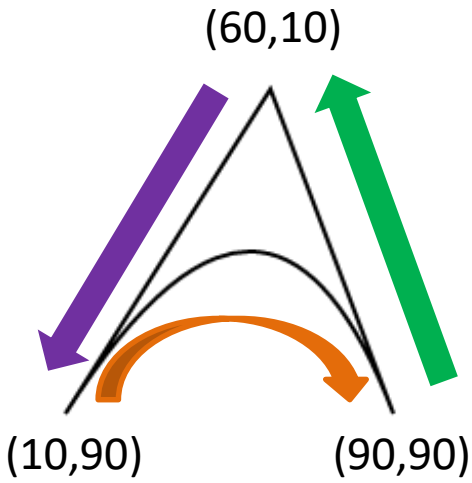
- A path may also contain curved lines.
- These are unfortunately a bit more involved.



- The quadraticCurveTo method draws a *quadratic curve* to a Destination point.
- To determine the curvature of the line, the method is given a Control point as well as a Destination point.
- Imagine this Control point as attracting the line, giving it its curve.
- The line won't go through the Control point, but its direction at the Start and Destination points will be such that a straight line in that direction would point toward the Control point.

Quadratic Curves

```
<canvas></canvas>
<script>
let cx = document.querySelector("canvas").getContext("2d");
cx.beginPath();
cx.moveTo(10, 90);
// control=(60,10) goal=(90,90)
cx.quadraticCurveTo(60, 10, 90, 90);
cx.lineTo(60, 10);
cx.closePath();
cx.stroke();
</script>
```

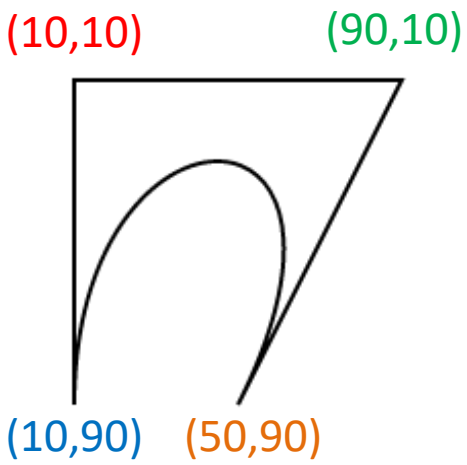


- This code draws a quadratic curve from the left (10,90) to the right (90,90), with (60,10) as the control point.
- Then draws two line segments going through that control point (60,10) and back to the start of the line (10,90).
- The result somewhat resembles a *Star Trek* insignia.
- You can see the effect of the control point: the lines leaving the lower corners start off in the direction of the control point and then curve toward their target.

Bézier (BEH-zee-ay) Curves

- The bezierCurveTo method draws a similar kind of curve called a Bézier curve.
- Instead of a single control point, this one has two—one for each of the line's endpoints.
- The two control points specify the direction at both ends of the curve.
- The farther they are away from their corresponding point, the more the curve will "bulge" in that direction.
- Here is an example of a Bézier curve:

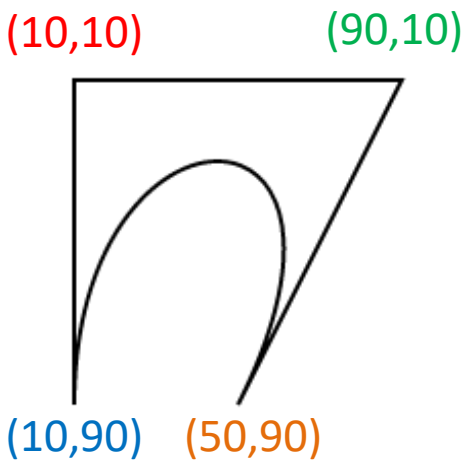
```
<canvas></canvas>
<script>
let cx = document.querySelector("canvas").getContext("2d");
cx.beginPath();
cx.moveTo(10, 90);
// control1=(10,10) control2=(90,10) goal=(50,90)
cx.bezierCurveTo(10, 10, 90, 10, 50, 90);
cx.lineTo(90, 10);
cx.lineTo(10, 10);
cx.closePath();
cx.stroke();
</script>
```



Bézier Curves

- Such curves can be hard to work with.
- It's not always clear how to find the control points that provide the shape you are looking for.
- Sometimes you can compute them.
- And sometimes you'll just have to find a suitable value by trial and error.

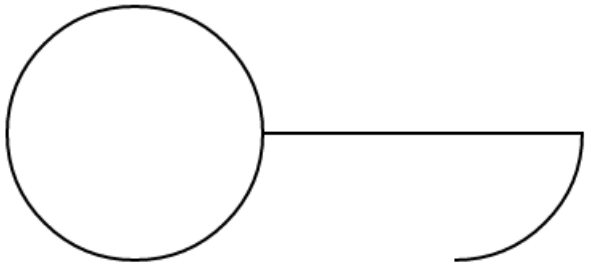
```
<canvas></canvas>
<script>
let cx = document.querySelector("canvas").getContext("2d");
cx.beginPath();
cx.moveTo(10, 90);
// control1=(10,10) control2=(90,10) goal=(50,90)
cx.bezierCurveTo(10, 10, 90, 10, 50, 90);
cx.lineTo(90, 10);
cx.lineTo(10, 10);
cx.closePath();
cx.stroke();
</script>
```



Arc Curves

- The arc method is a way to draw a line that curves along the edge of a circle.
- It takes a pair of coordinates for the arc's center, a radius, and then a start angle and end angle.
- Those last two parameters make it possible to draw only part of the circle.
- The angles are measured in radians, not degrees.
- This means a full circle has an angle of 2π, or 2 * Math.PI, which is about 6.28.
- The angle starts counting at the point to the right of the circle's center and goes clockwise from there.
- You can use a start of 0 and an end bigger than 2π (say, 7) to draw a full circle.

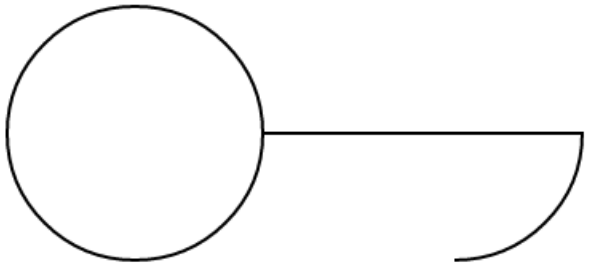
```
<canvas></canvas>
<script>
let cx = document.querySelector("canvas").getContext("2d");
cx.beginPath();
// center=(50,50) radius=40 angle=0 to 7
cx.arc(50, 50, 40, 0, 7);
// center=(150,50) radius=40 angle=0 to ½π
cx.arc(150, 50, 40, 0, 0.5 * Math.PI);
cx.stroke();
</script>
```



Arc Curves

- This code draws a picture containing a line from the right of the full circle (first call to arc) …
- to the right of the quarter-circle (second call).
- Like other path-drawing methods, a line drawn with arc is connected to the previous path segment.
- You can call moveTo or start a new path to avoid this.

```
<canvas></canvas>
<script>
let cx = document.querySelector("canvas").getContext("2d");
cx.beginPath();
// center=(50,50) radius=40 angle=0 to 7
cx.arc(50, 50, 40, 0, 7);
// center=(150,50) radius=40 angle=0 to ½π
cx.arc(150, 50, 40, 0, 0.5 * Math.PI);
cx.stroke();
</script>
```

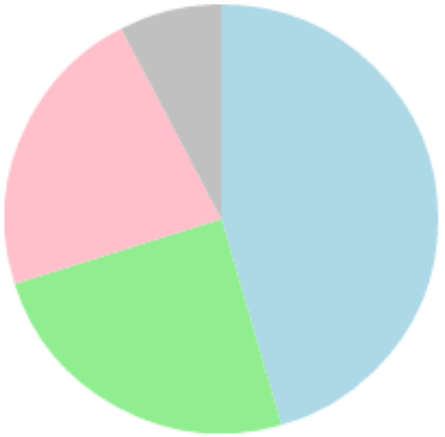


Any Questions?

Drawing a Pie Chart

- Imagine you've just taken a job at EconomiCorp, Inc.
- Your first assignment is to draw a pie chart of its customer satisfaction survey results.
- The results binding contains an array of objects that represent the survey responses.

```
const results = [
{name: "Satisfied", count: 1043, color: "lightblue"},
{name: "Neutral", count: 563, color: "lightgreen"},
{name: "Unsatisfied", count: 510, color: "pink"},
{name: "No comment", count: 175, color: "silver"}
];
```

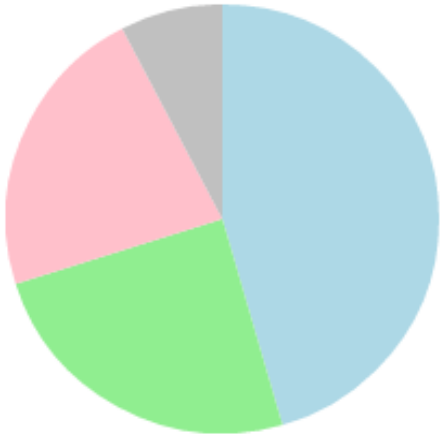


- To draw a pie chart, we draw a number of pie slices.
- Each slice made up of an arc and a pair of lines to the center of that arc.
- We can compute the angle taken up by each arc by dividing a full circle (2π) by the total number of responses …
- and then multiplying that number (the angle per response) by the number of people who picked a given choice.

Drawing a Pie Chart

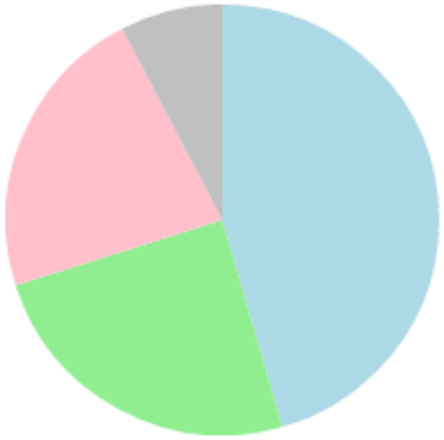
```
const results = [
{name: "Satisfied", count: 1043, color: "lightblue"},
{name: "Neutral", count: 563, color: "lightgreen"},
{name: "Unsatisfied", count: 510, color: "pink"},
{name: "No comment", count: 175, color: "silver"}
];
```

```
<canvas width="200" height="200"></canvas>
<script>
  let cx = document.querySelector("canvas").getContext("2d");
  let total = results
  .reduce((sum, {count}) => sum + count, 0);
  // Start at the top
  let currentAngle = -0.5 * Math.PI;
  for (let result of results) {
     let sliceAngle = (result.count / total) * 2 * Math.PI;
     cx.beginPath();
     // center=100,100, radius=100
     // from current angle, clockwise by slice's angle
     cx.arc(100, 100, 100,
     currentAngle, currentAngle + sliceAngle);
     currentAngle += sliceAngle;
     cx.lineTo(100, 100);
     cx.fillStyle = result.color;
     cx.fill();
  }
</script>
```



Any Questions?

Text



- But a chart that doesn't tell us what the slices mean isn't very helpful.
- We need a way to draw text to the canvas.

- A 2D canvas drawing context provides the methods fillText and strokeText.
- strokeText can be useful for outlining letters.
- But usually fillText is what you need.
- It will fill the outline of the given text with the current fillStyle.

```
<p>Normal HTML before.</p>
<canvas width="200" height="200"></canvas>
<script>
 let cx = document.querySelector("canvas").getContext("2d");
 cx.font = "28px Georgia";
 cx.fillStyle = "fuchsia";
 cx.fillText("I can draw text,", 10, 50);
</script>
<p>Normal HTML after.</p>
```

Normal HTML before.

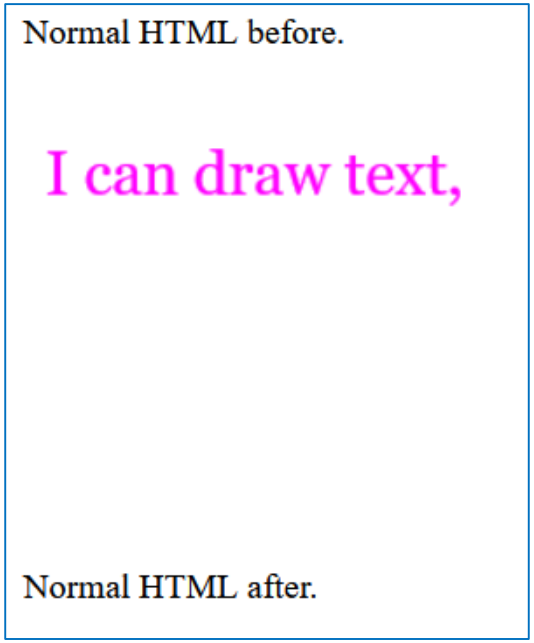
I can draw text,

Normal HTML after.

Text

- You can specify the size, style, and font of the text with the font property.
- This example just gives a font size (28px) and family name (Georgia).
- It is also possible to add italic or bold to the start of the string to select a style.

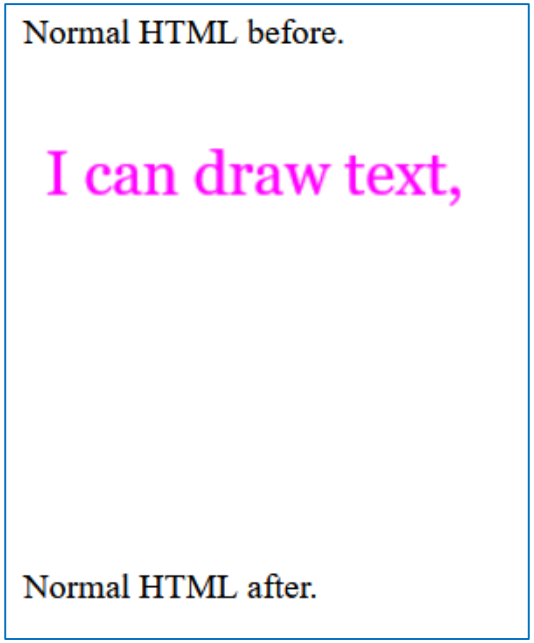
```
<p>Normal HTML before.</p>
<canvas width="200" height="200"></canvas>
<script>
 let cx = document.querySelector("canvas").getContext("2d");
 cx.font = "28px Georgia";
 cx.fillStyle = "fuchsia";
 cx.fillText("I can draw text,", 10, 50);
</script>
<p>Normal HTML after.</p>
```



Text

- The last two arguments to fillText and strokeText provide the position at which the font is drawn.
- By default, they indicate the position of the start of the text's alphabetic baseline.
- The baseline is the line that letters "stand" on, …
- not counting hanging parts in letters such as j or p.

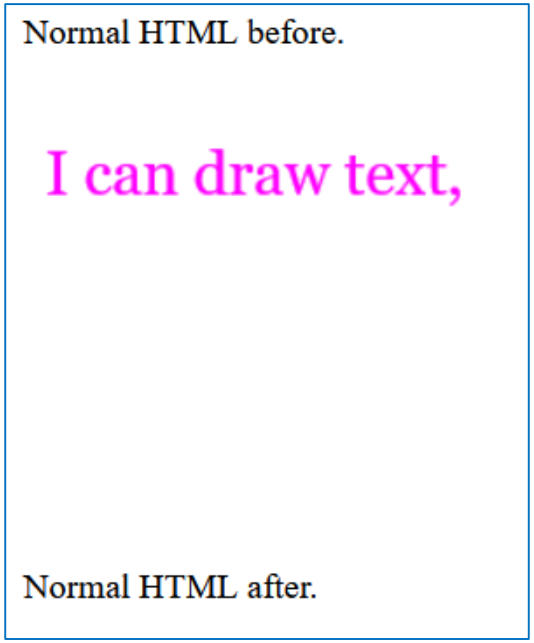
```
<p>Normal HTML before.</p>
<canvas width="200" height="200"></canvas>
<script>
 let cx = document.querySelector("canvas").getContext("2d");
 cx.font = "28px Georgia";
 cx.fillStyle = "fuchsia";
 cx.fillText("I can draw text,", 10, 50);
</script>
<p>Normal HTML after.</p>
```



Text

- You can change the horizontal position by setting the textAlign property to "end" or "center".
- And the vertical position by setting textBaseline to "top", "middle", or "bottom".
- In Assignment 4, you will come back to our pie chart to solve the problem of labeling the slices.

```
<p>Normal HTML before.</p>
<canvas width="200" height="200"></canvas>
<script>
 let cx = document.querySelector("canvas").getContext("2d");
 cx.font = "28px Georgia";
 cx.fillStyle = "fuchsia";
 cx.fillText("I can draw text,", 10, 50);
</script>
<p>Normal HTML after.</p>
```



Any Questions?

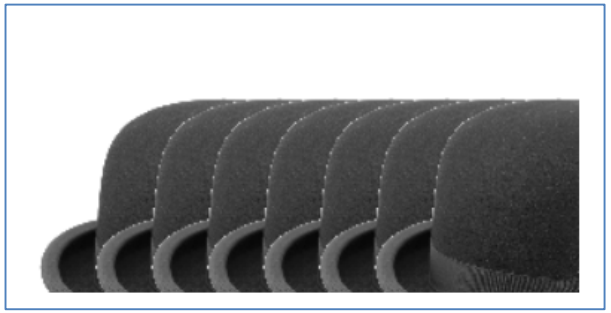
Images

- In computer graphics, a distinction is often made between vector graphics and bitmap graphics.
- Vector graphics is what we have been doing so far:
  - specifying a picture by giving a logical description of shapes.
- Bitmap graphics don't specify actual shapes but rather:
  - work with pixel data, i.e., rasters of colored dots.
- The drawImage method allows us to draw pixel data onto a canvas.
- This pixel data can originate from an <img> element or from another canvas.

Images

- The following example creates a detached <img> element and loads an image file into it.
- But it cannot immediately start drawing from this picture because the browser may not have loaded it yet.
- To deal with this, we register a "load" event handler and do the drawing after the image has loaded.

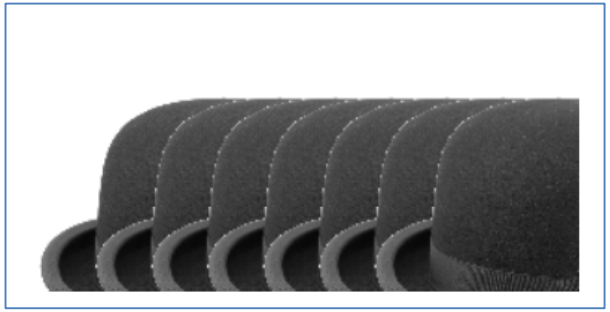
```
<canvas></canvas>
<script>
  let cx = document.querySelector("canvas").getContext("2d");
  let img = document.createElement("img");
  img.src = "img/hat.png";
  img.addEventListener("load", () => {
    for (let width = 10; width < 200; width += 30) {
    cx.drawImage(img, width, 10);
    }
  });
</script>
```



Images

- By default, drawImage will draw the image at its original size.
- You can also give it two additional arguments to set a different width and height.
- This code will draw the image of a hat 7 times.
- The height of each hat will be 10 pixels.
- The width will be 10, 40, 70, 100, 130, 160, and 190 pixels

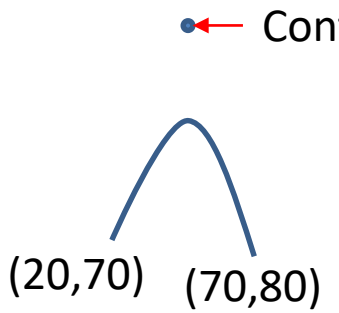
```
<canvas></canvas>
<script>
  let cx = document.querySelector("canvas").getContext("2d");
  let img = document.createElement("img");
  img.src = "img/hat.png";
  img.addEventListener("load", () => {
    for (let width = 10; width < 200; width += 30) {
    cx.drawImage(img, width, 10);
    }
  });
</script>
```



Any Questions?

In-Class Problem

1. Write a canvas script that draws this quadrature curve with a control point of (60,10).



2. Write a canvas script that draws a Bézier curve from (20,90) to (60,90) with control points of (20,10) and (100,10); then draws lines from the destination point to the right-most control point, back to the left-most control point, and then to the starting point; and finally fills in the shape.