CSC 240
Computer Graphics

Sara Mathieson
Fall 2016
Smith College

Outline: 10/17

- Recap Homework 4
- Finish Curves
- Line clipping algorithm
- Lab 5
  - **HW 5** due Monday Oct 24
  - **Office Hours** today 4-5pm (Ford 015)
  - **Midterm**
    - Out Tuesday Oct 25
    - Due late Fri / early Sat (Oct 28/29)
    - Young Science Library
    - 2 hours + small coding question with unlimited time

Recap Homework 4:

Live Coding Demo!

Matrix Multiply

- Testing: try an example that's more challenging than immediately needed
- Your code might work on simple tests, but actually doesn't work in general
- Build up loops from the inside out

3x2

$$\begin{bmatrix} 5 & 1 \\ -2 & 3 \\ 6 & 0 \end{bmatrix}$$

2x3

$$\begin{bmatrix} -1 & 2 & 3 \\ 0 & 4 & 1 \end{bmatrix}$$

= ?

Matrix Multiply

- Testing: try an example that's more challenging than immediately needed
- Your code might work on simple tests, but actually doesn't work in general
- Build up loops from the inside out

3x2

$$\begin{bmatrix} 5 & 1 \\ -2 & 3 \\ 6 & 0 \end{bmatrix}$$

2x3

$$\begin{bmatrix} -1 & 2 & 3 \\ 0 & 4 & 1 \end{bmatrix}$$

= ?

3x3

Matrix Multiply: Loop 1

```
function matrixMultiply(M1, M2) {

    var K = M2.length;

    var s = 0;
    for (var k=0; k < K; k++) {
        s += M1[0][k]*M2[k][0]; // match inner dimension indices
    }
    console.log(s)
}
```

This loop only computes the first entry!

$$\overbrace{\begin{bmatrix} 5 & 1 \\ -2 & 3 \\ 6 & 0 \end{bmatrix}}^{K} \begin{bmatrix} -1 & 2 & 3 \\ 0 & 4 & 1 \end{bmatrix} \Big\}K = \begin{bmatrix} -5 & \\ & \\ & \end{bmatrix}$$

Matrix Multiply: Loop 2

```
function matrixMultiply(M1, M2) {

    var K = M2.length; // # rows of M2
    var J = M2[0].length; // # columns of M2

    var row = [];
    for (var j=0; j < J; j++) {
        var s = 0;
        for (var k=0; k < K; k++) {
            s += M1[0][k]*M2[k][j]; // match inner dimension indices
        }
        row.push(s);
    }
    console.log(row)
}
```

This loop computes the first row

$$\begin{bmatrix} 5 & 1 \\ -2 & 3 \\ 6 & 0 \end{bmatrix} \overbrace{\begin{bmatrix} -1 & 2 & 3 \\ 0 & 4 & 1 \end{bmatrix}}^{\mathrm{J}} = \begin{bmatrix} -5 & 14 & 16 \\ & & \\ & & \end{bmatrix}$$

Matrix Multiply: Loop 3

```
function matrixMultiply(M1, M2) {
    var I = M1.length;     // # rows of M1
    var K = M2.length;     // # rows of M2 = # cols of M1
    var J = M2[0].length; // # columns of M2

    var result = [];
    for (var i=0; i < I; i++) {
        var row = [];
        for (var j=0; j < J; j++) {
            var s = 0;
            for (var k=0; k < K; k++) {
                s += M1[i][k]*M2[k][j]; // match inner dimension indices
            }
            row.push(s);
        }
        result.push(row);
    }
    return result;
}
```

$$\overset{\overbrace{}^{J}}{\begin{bmatrix} -1 & 2 & 3 \\ 0 & 4 & 1 \end{bmatrix}}$$

$$I \left\{ \begin{bmatrix} 5 & 1 \\ -2 & 3 \\ 6 & 0 \end{bmatrix} \right. \begin{bmatrix} -1 & 2 & 3 \\ 0 & 4 & 1 \end{bmatrix} = \begin{bmatrix} -5 & 14 & 16 \\ 2 & 8 & -3 \\ -6 & 12 & 18 \end{bmatrix}$$

Recap Homework 4:

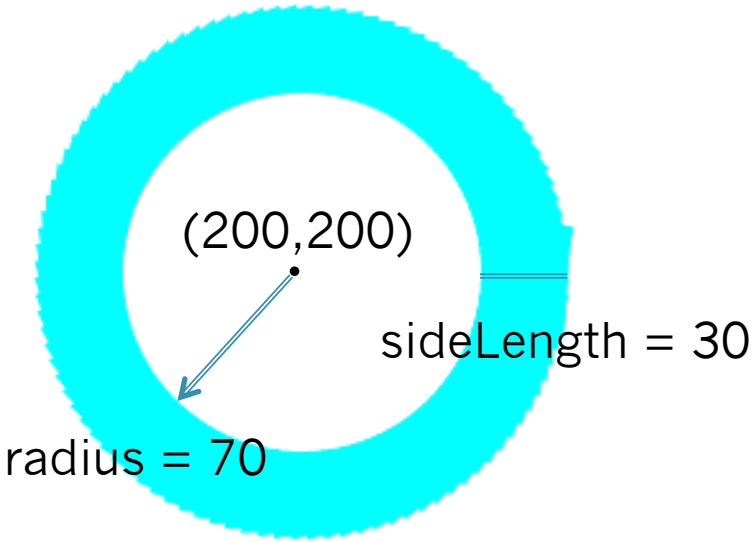
Figure 8

Figure 8 Setup

```
// variables to keep track of when we need to switch the figure 8
var counter = 1;
var circle = 100; // number of times to loop for each circle
var angle = 2*Math.PI/circle; // angle to rotate each time

// figure 8 parameters
var sideLength = 30; // side length of our square
var radius = 70;     // radius
```

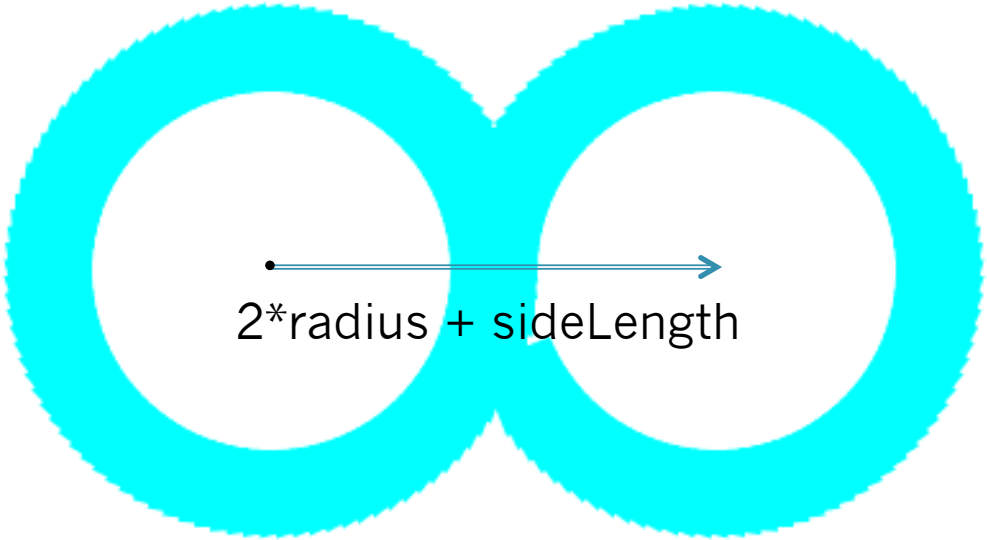
```
function draw() {
    // testing
    //scale(2,3);
    //rotate(0.1);
    //translate(30,40);
    //reflect();
    //shear(2);

    // set fill color
    graphics.fillStyle = "aqua";

    // transformations we want only once
    translate(200,200)
}
```

Figure 8



```
function figure8() {

    // first circle of the figure 8
    if (counter <= circle) {
        rotate(-angle);
        graphics.fillRect(radius,0,sideLength,sideLength);

        // when we have completed half the figure 8, shift over
        if (counter == circle) {
            translate(2*radius+sideLength+4,0); // +4 makes it look a bit smoother
        }

    // second circle of the figure 8
    } else {
        rotate(angle);
        graphics.fillRect(-(radius+sideLength),0,sideLength,sideLength);

        // when we have reached 2 loops, reset
        if (counter == 2*circle) {
            translate(-(2*radius+sideLength+4),0); // +4 makes it look a bit smoother
            counter = 0; // reset counter to 0

            // clear background
            graphics.save();
            graphics.setTransform(1, 0, 0, 1, 0, 0);
            graphics.clearRect(0, 0, canvas.width, canvas.height);
            graphics.restore();
        }
    }

    // increment counter
    counter += 1;
}
```

Figure 8



DEMO

```
function figure8() {

    // first circle of the figure 8
    if (counter <= circle) {
        rotate(-angle);
        graphics.fillRect(radius,0,sideLength,sideLength);

        // when we have completed half the figure 8, shift over
        if (counter == circle) {
            translate(2*radius+sideLength+4,0); // +4 makes it look a bit smoother
        }

    // second circle of the figure 8
    } else {
        rotate(angle);
        graphics.fillRect(-(radius+sideLength),0,sideLength,sideLength);

        // when we have reached 2 loops, reset
        if (counter == 2*circle) {
            translate(-(2*radius+sideLength+4),0); // +4 makes it look a bit smoother
            counter = 0; // reset counter to 0

            // clear background
            graphics.save();
            graphics.setTransform(1, 0, 0, 1, 0, 0);
            graphics.clearRect(0, 0, canvas.width, canvas.height);
            graphics.restore();
        }
    }

    // increment counter
    counter += 1;
}
```