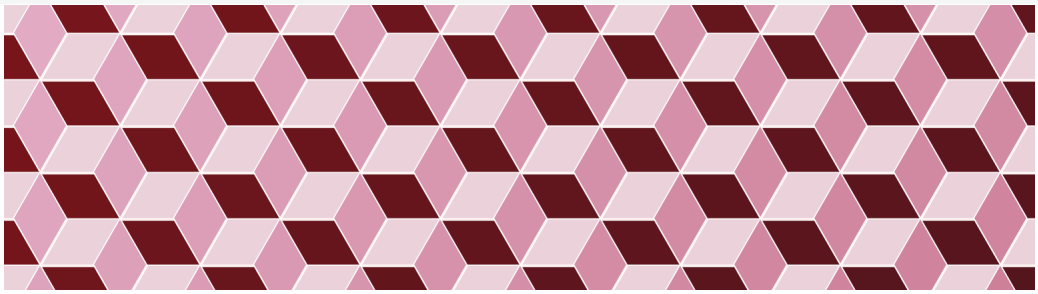


15-104 Introduction to Computing for Creative Practice

Fall 2022

11 More Functions

Instructor: Tom Cortina, tcortina@cs.cmu.edu, GHC 4117, 412-268-3514

3

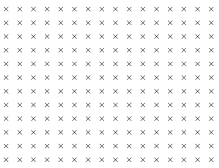
Functions

- Functions are used to collect instructions together that perform a specific task.
- Functions compartmentalize the code into individually managed units which can be debugged/managed separately.
- Functions manage complexity of your code by "hiding" finer details to help the programmer focus on the overall design task.
  - `random`, `ellipse`, `translate`
  - "functional abstraction"

4

Five Options (Casey Reas and Chandler McWilliam)

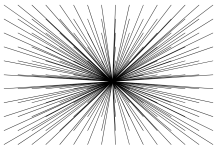
```
var option = 1;
function setup() {
    createCanvas(640, 480);
    noFill();
    frameRate(5); // reduce computation
}
function draw() {
    background(255);
    var density = map(mouseX, 0, width, 20, 50);
    if (option == 1) { // Option 1: Stitches
        for (var x = 50; x <= width-50; x += density) {
            for (var y = 50; y <= height-50; y+=density) {
                line(x-5, y-5, x+5, y+5);
                line(x+5, y-5, x-5, y+5);
            }
        }
```

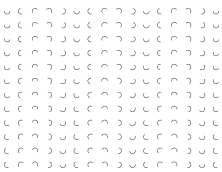


5

Five Options (cont'd)

```
    } else if (option == 2) { // Option 2: Perspective
        for (var x = 50; x <= width-50; x += density) {
            for (var y = 50; y <= height-50; y+=density) {
                line(x, y, width/2, height/2);
            }
        }
    } else if (option == 3) { // Option 3: Rotating arcs
        var count = 120;
        for (var x = 50; x <= width-50; x += density) {
            for (var y = 50; y <= height-50; y+=density) {
                var s = map(count, 120, 0, 0, TWO_PI*2);
                arc(x, y, 14, 14, s, s + PI);
                count--;
            }
        }
```

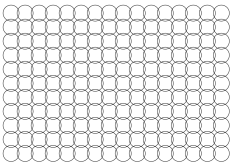




6

Five Options (cont'd)

```
    } else if (option == 4) { // Option 4: Overlapping circles
        for (var x = 50; x <= width-50; x += density) {
            for (var y = 50; y <= height-50; y+=density) {
                ellipse(x, y, 40, 40);
            }
        }
```



7

Five Options (cont'd)

```
    } else if (option == 5) { // Option 5: Groups of five
        for (var x = 50; x < width-50; x += density) {
            for (var y = 50; y < height-50; y+=density) {
                for (var i = 0; i < 16; i+=4) {
                    line(x + i, y, x + i, y + 12);
                }
                line(x, y, x + 12, y + 12);
            }
        }
    }
}
function mousePressed() {
   option++;
   if (option > 5) { option = 1; }
}
```

8

Five Options with functions

```
var option = 0;
var density;        // so all functions can access this value
function setup() {
    createCanvas(640, 480);
    noFill();
    frameRate(5); // reduce computation
}
function draw() {
    background(255);
    density = map(mouseX, 0, width, 20, 50);
    drawOption(option);
}
function mousePressed() {
   option = (option + 1) % 5;     // 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, ...
}
```

9

Five Options with functions (cont'd)

```
function drawOption(option) {
    if (option == 0) { stitches(); }
    else if (option == 1) { line_perspective(); }
    else if (option == 2) { rotating_arcs(); }
    else if (option == 3) { overlapping_circles(); }
    else if (option == 4) { groups_of_five(); }
}

function stitches() {
    for (var x = 50; x <= width-50; x += density) {
        for (var y = 50; y <= height-50; y+=density) {
            line(x-5, y-5, x+5, y+5);
            line(x+5, y-5, x-5, y+5);
        }
    }
}
```

10

Five Options with functions (cont'd)

```
function line_perspective() {
    for (var x = 50; x <= width-50; x += density) {
        for (var y = 50; y <= height-50; y+=density) {
            line(x, y, width/2, height/2);
        }
    }
}
function rotating_arcs() {
    var count = 120;
    for (var x = 50; x <= width-50; x += density) {
        for (var y = 50; y <= height-50; y+=density) {
            var s = map(count, 120, 0, 0, TWO_PI*2);
            arc(x, y, 14, 14, s, s + PI); count--;
        }
    }
}
```

11

Five Options with functions (cont'd)

```
function overlapping_circles() {
    for (var x = 50; x <= width-50; x += density) {
        for (var y = 50; y <= height-50; y+=density) {
            ellipse(x, y, 40, 40);
        }
    }
}
function groups_of_five() {
    for (var x = 50; x <= width-50; x += density) {
        for (var y = 50; y <= height-50; y+=density) {
            for (var i = 0; i < 16; i+=4) { line(x + i, y, x + i, y + 12); }
            line(x, y, x + 12, y + 12);
        }
    }
}
```

12

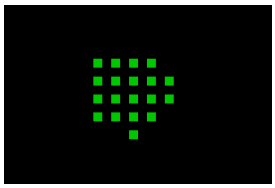
The `return` statement

- The purpose of some functions is to compute and return an answer to another part of the program to use.

  e.g.
  ```
  var x = random(15, 104);
  z = 15 + sqrt(104);
  ```
- In your own functions, you can return a value with the `return` statement.
- Caution: Once a `return` statement executes, flow of control goes back to the calling function immediately, even if there are more instructions after the `return` statement!

13

Example: `nearMouse`



```
function setup() {
    createCanvas(300, 200);
    noStroke();
    frameRate(5);
}
function draw() {
    background(0);
    for (var y = 0; y < height; y += 20) {
        for (var x = 0; x < width; x += 20) {
            fill(0, 200, 0);
            if (nearMouse(x, y) === true) { rect(x, y, 10, 10); }
        }
    }
}
```

Another way:
```
if (nearMouse(x, y)) { rect(x, y, 10, 10); }
```

14

Example: `nearMouse` (cont'd)

```
function nearMouse(x, y) {
    if (dist(x, y, mouseX, mouseY) < 50) {
        return true;
    }
    return false;
}
```

Another way:

```
function nearMouse(x, y) {
    return dist(x, y, mouseX, mouseY) < 50;
}
```

15

Try This

- Modify the previous example so that `nearMouse` takes a 3rd parameter which is the distance threshold (which is currently fixed at 50). Make a global variable `distThreshold` in the program and initialize it to a new distance threshold, say, 30. Use the global variable `distThreshold` as the third parameter in the call to `nearMouse`.
- Suppose that instead of passing a third parameter to `nearMouse` you just compared the distance to `distThreshold` and did not add a third parameter.
  - Do you think this would this be a better solution?
  - When would this not be a better solution?

16