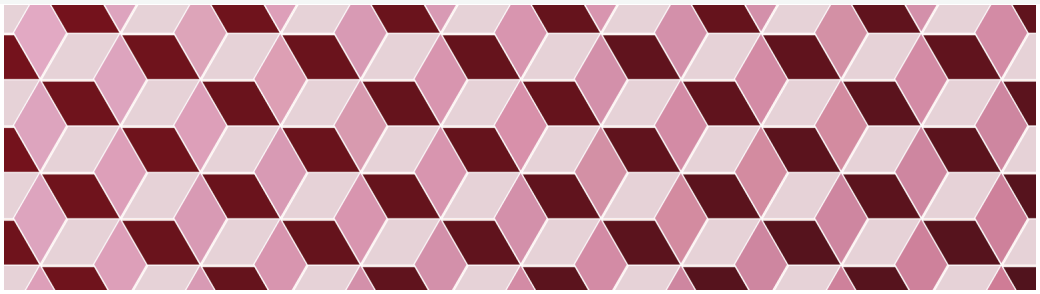


15-104 Introduction to Computing for Creative Practice

Fall 2020

13 Arrays

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

1

Arrays

- An array with n elements (n > 0) is an ordered collection of values of the same type, indexed from 0 to n-1. (ordered does not necessarily mean sorted here)
- e.g. an array holding noon temperatures in Fahrenheit for the past week might be stored as:

```
temps = [79, 81, 57, 64, 63, 57, 57]
```

- Arrays should not hold data values of different types.
- Remember that an array is zero-indexed.
- To access an array, we use "subscript" notation:

```
print temps[1];
last_three_day_average = (temps[4] + temps[5] + temps[6]) / 3;
```

2

Arrays

- Since arrays are indexed with integers, we can write loops to access every element of the array easily:

```
sum = 0;
for (var i = 0; i < temps.length; i++) {
    sum += temps[i];
average = sum / temps.length;

min = temps[0];
for (var j = 1; j < temps.length; j++) {
    if (temps[j] < min) { min = temps[j]; }
}
```

3

One Box

`color()` creates colors for storing in variables based on the color mode set by `colorMode()` (default: RGB).


```
var x;
var y;
var dx;
var dy;
var c;
function setup() {
    createCanvas(200, 200);
    x = random(width);
    y = random(height);
    dx = random(-5, 5);
    dy = random(-5, 5);
    c = color(random(255),
       random(255), random(255));
    frameRate(5);
}
```

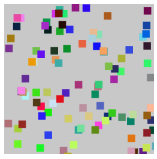
```
function draw() {
    background(200, 200, 200);
    noStroke();
    fill(c);
    rect(x, y, 10, 10);
    x += dx;
    y += dy;

    if (x > width)  { x = 0; }
    else if (x < 0) { x = width; }

    if (y > height) { y = 0; }
    else if (y < 0) { y = height; }
}
```

4

Many Boxes using Arrays

empty array

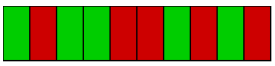
```
var x = [];
var y = [];
var dx = [];
var dy = [];
var c = [];
function setup() {
    createCanvas(200, 200);
    for (var i=0; i<100; i++)  {
        x[i] = random(width);
        y[i] = random(height);
        dx[i] = random(-5, 5);
        dy[i] = random(-5, 5);
        c[i] = color(random(255),
          random(255), random(255));
    }
    frameRate(5);
}
```



```
function draw() {
    background(200, 200, 200);
    noStroke();
    for (i=0; i<100; i++) {
        fill(c[i]);
        rect(x[i], y[i], 10, 10);
        x[i] += dx[i];
        y[i] += dy[i];
        if (x[i] > width)
                    { x[i] = 0; }
        else if (x[i] < 0)
                    { x[i] = width; }
        if (y[i] > height)
                    { y[i] = 0; }
        else if (y[i] < 0)
                    { y[i] = height; }
    }
}
```

5

Click the Buttons



```
var isRed = []

function setup() {
    createCanvas(201, 41);
    for (i = 0; i < 10; i++) {
        isRed[i] = false;
    }
    frameRate(5);
}

function draw() {
    background(255);
    // cont'd in next column
```

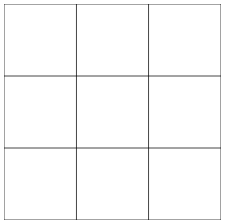
`int()` returns the integer value of the argument (removing the fractional part)

```
    for (i = 0; i < 10; i++) {
        // for each button...
        if (isRed[i]) {
            // is the button i red?
            fill(205, 0, 0); // red
        } else {
            fill(0, 205, 0); // green
        } // draw the button i
            rect(i * 20, 0, 20, 40);
    }
}

function mousePressed() {
    var index = int(mouseX / 20);
    isRed[index] = ! isRed[index];
}
```

6

Tic Tac Toe

```
var size;   // size of each square
var BLANK = "white";  // the "empty" state value
var state = [BLANK, BLANK, BLANK, BLANK, BLANK, BLANK, BLANK, BLANK, BLANK];
var turn = "red"; // whose turn is it? (red or green)
var gameOver = false;

// state layout is like this:
// 0 1 2
// 3 4 5
// 6 7 8

function setup() {
    createCanvas(300, 300);
    size = width / 3;
}
```



7

Tic Tac Toe (cont'd)

```
function mouseInsideSquare(x, y, w) {
    return mouseX > x && mouseX < x + w && mouseY > y && mouseY < y + w;
}

function mouseToIndex() {
    var x = 0;
    var y = 0;
    for (var index = 0; index < 9; index++) {
        if (mouseInsideSquare(x, y, size)) { return index; }
        x += size;
        if (x >= width) { x = 0; y += size; }
    }
    return -1;
}
```

Tests if mouse click is inside each square.
If so, it immediately returns that square's index.
If mouse is not clicked in any of the squares,
it returns -1 after all squares are checked.

8

Tic Tac Toe (cont'd)

```
function mousePressed() {
    var index = mouseToIndex();  // from previous slide
    if (!gameOver && index != -1 && state[index] == BLANK) {
        state[index] = turn;
        if (turn == "red") {
            turn = "green";
        } else {
            turn = "red";
        }
    }
}
```

The order matters here.
Why?

9

Tic Tac Toe (cont'd)

Remember: if a `return` instruction is executed, the function returns immediately to the calling function and does not continue with the rest of its code.

```
function checkWin() {
    // check for 3 in a row horizontally
    for (var i = 0; i < 9; i += 3) {
        if (state[i] == state[i + 1] && state[i] == state[i + 2]
            && state[i] != BLANK) { // winner cannot be "BLANK"!
                return state[i]; // a winner
        }
    }
    // check for 3 in a row vertically
    for (var j = 0; j < 3; j += 1) {
        if (state[j] == state[j + 3] && state[j] == state[j + 6]
            && state[j] != BLANK) { // winner cannot be "BLANK"!
                return state[j]; // a winner
        }
    }
```

10

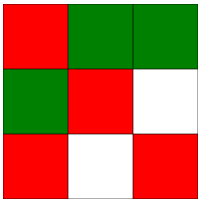
Tic Tac Toe (cont'd)

```
    // function checkWin() cont'd

    // check for diagonal wins
    if (state[0] == state[4] && state[0] == state[8]
        && state[0] != BLANK) { // winner cannot be "BLANK"!
            return state[0]; // from top left to bottom right
    }
    if (state[6] == state[4] && state[6] == state[2]
        && state[6] != BLANK) { // winner cannot be "BLANK"!
            return state[6]; // from bottom left to top right
    }

    return "none"; // no winning 3-in-a-row
}
```

11

Tic Tac Toe (cont'd)

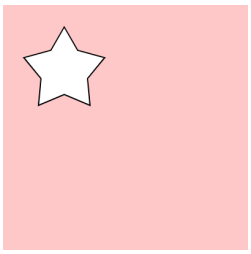
```
function draw() {           // the game!
    background(220);
    var index = 0;          // draw the game board with current state (colors)
    for (var row = 0; row < 3; row++) {
        for (var column = 0; column < 3; column++) {
            fill(state[index]);
            rect(column*size, row*size, size, size);
            index = index + 1;
        }
    }
    if (checkWin() != "none") {
        print("Game Over!");
        gameOver = true;
        noLoop(); // we are done
    }
}
```



12

Star

```
var x = [50, 61, 83, 69, 71, 50, 29, 31, 17, 39];
var y = [18, 37, 43, 60, 82, 73, 82, 60, 43, 37];

function setup() {
    createCanvas(200, 200);
}

function draw() {
    background(255, 200, 200);
    var nPoints = x.length;
    beginShape();
    for (var i = 0; i < nPoints; i++) {
        vertex(x[i], y[i]);
    }
    endShape(CLOSE);
    noLoop();
}
```

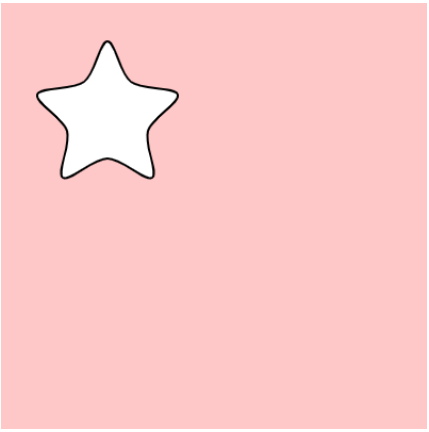


You can create arbitrary shapes by using the `beginShape` function, then create a series of vertices using the `vertex` function, and then end the shape with the `endShape` function.

13

Try This

Read how `curveVertex()` works, and draw the original star with curved sections rather than straight sections.



14