introduction to media computing
week 07

School of creative media, mw

Today’s topics (week 07)

- quick review
- Arrays Part 2 - array related functions

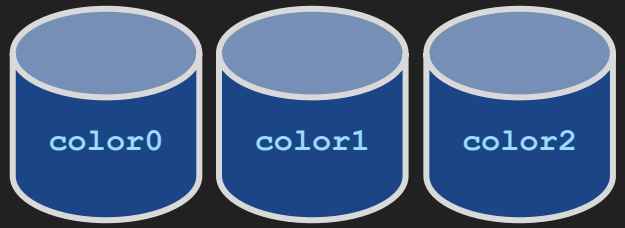
Today's topics (week 07)

- quick review
- Arrays Part 2 - array related functions

- utility function `map()`

Review: Array

3 Simple variables




```
let color0 = 255;
let color1 = 200;
let color2 = 150;
```

An ARRAY with 3 members

Each member is accessed by its index

```
let colors = [];
colors[0] = 255;
colors[1] = 200;
colors[2] = 150;
```

Review: Array

```
let activeColor = 0;
let color0 = 255;
let color1 = 200;
let color2 = 150;

if (activeColor == 0) {
  fill(color0);
}
else if (activeColor == 1) {
  fill(color1);
}
else if (activeColor == 2) {
  fill(color2);
}
```

```
let activeColor = 0;
let colors = [];
colors[0] = 255;
colors[1] = 200;
colors[2] = 150;

fill( colors[activeColor] );
```

When `activeColor` equals to 1, then we are filling with `colors[1]`.

Review: Array

To declare an array:

```
let posX = [];
```

To declare an array and initialize it at the same time:

```
let posX = [10,20,30];
```

Review: `lerp(start,stop,amount)`

`lerp()` computes a number between the `'start'` and `'end'` values determined by the given `'amount'` (a value between 0.0 to 1.0).

Examples:

`lerp(0, 10, 0.0)` returns `0`

`lerp(0, 10, 1.0)` returns `10`

`lerp(0, 10, 0.25)` returns `2.5`

`lerp(0, 10, 0.5)` returns `5`

[ ] Array Part 2
Array property and related functions

Array: the property `.length`

A convenient 'property' which tells us the number of members in the array.

```
let posX = [10,20,30];
let n = posX.length;
//  n equals to 3.
```

Array: the property `.length`

As an array can grow any time, it is useful for defining loop conditions.

```
let posX = [10,20,30,40,50,60];
for (let i = 0; i < posX.length; i++) {
  // some code here
}
```

Array: add a new element by `.push()`

Adds the given data to the end of the array

```
let posX = [10,20,30];
posX.push(70);
// posX = [10,20,30,70]
```

Array: add a new element by `.pop()`

Removes the last element from the array

```
let posX = [10,20,30];
posX.pop();
// posX = [10,20]
```

Array: partially copy an array by `.slice()`

`.slice(start,end)` copies and returns the elements from the `start` index, and up to (but not including) the `end` index.

```
let posX = [10,20,30,40,50,60];
let someX = posX.slice(1,3);
// someX = [20,30]
```

Array: sorting by `.sort()`

`.sort()` by default sorts the array alphabetically, NOT numerically.

Example A

```
let names = ['Roy','Ann','Sam'];
names.sort();
// names = ['Ann','Roy','Sam']
```

Array: sorting by `.sort()`

`.sort()` by default sorts the array alphabetically, NOT numerically.

Example B

```
let numbers = [20,1000,300];
numbers.sort();
// numbers = [1000,20,300]
```

Array: `.sort()` numerically*

We have to provide a special 'compare function' to `.sort()` in order to sort numerically. We may use it as-is at this moment.

Example C

```
let numbers = [200,1000,30];
numbers.sort(function(a, b){return a - b});
// numbers = [30,200,1000]
```

Array Part 2: Summary

Array property

`.length`	returns number of elements

Array functions

`.push(A)`	appends `A` to the array
`.pop()`	removes the last element
`.slice(s,e)`	copies some elements
`.sort()`	sorts alphabetically

.push() and .sort() numerically

try p5*

JS

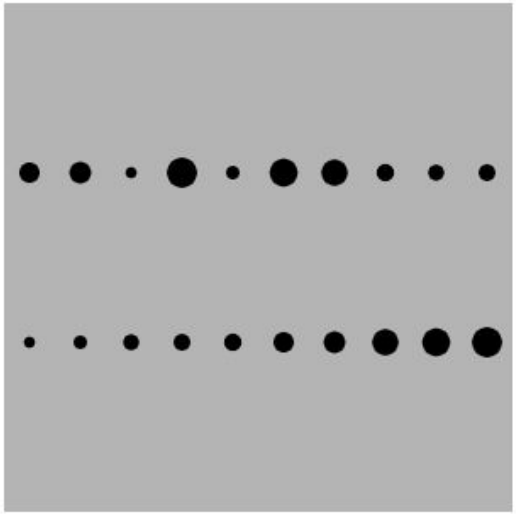
```
function setup(){
  createCanvas(300,300);
  background(180);
  noStroke();
  fill(0);

  //  our array of balls (sizes)
  let balls = [];
  //  push random numbers into our array
  for (let i = 0; i < 10; i++) {
    balls.push(random(20));
  }

  //  draw BEFORE sorting
  for (let i = 0; i < balls.length; i++) {
    ellipse(15 + i*30, 100, balls[i]);
  }
```

Result

EDIT ON CODEPEN

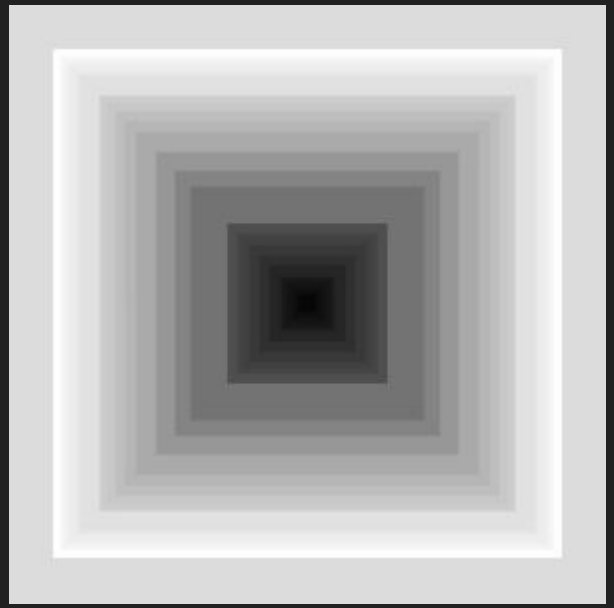


Resources 1x 0.5x 0.25x Rerun

canvas

exercise

In-class exercise 1



1. Use an array to store the size of the squares to be drawn.
2. For every 100ms, add a new square of random size to the array. (use the p5.js function `millis()` to keep track of time)
3. Draw the squares in your array as shown on the left
4. When the number of squares reaches 50, clear your array and repeat step 2.

Utility function `map()`

map(s, sStart, sEnd, dStart, dEnd)

p5*

`map()` maps a given number `s` within a range defined by `sStart` and `sEnd` to a new destination range defined by `dStart` and `dEnd`.

Example: `map(0.5, 0, 1, 0, 200)` returns `100`

Using p5.js `map()`

try p5*

A grid of rectangles which adapts to the mouse cursor position using `map()`.

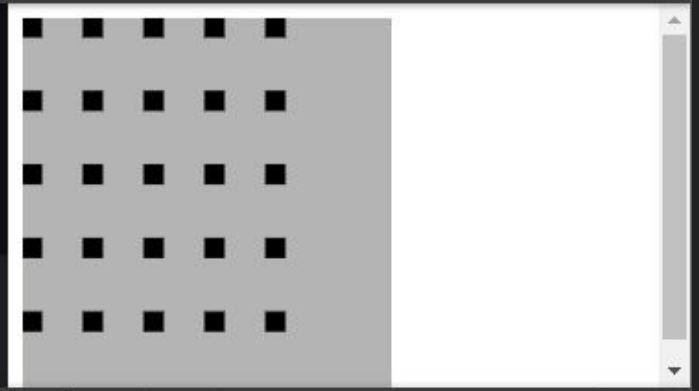
JS

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

function draw() {
  background(180);
  fill(0);
  for (let y = 0; y < 5; y++) {
    for (let x = 0; x < 5; x++) {
      //  remap (x,y) to a bigger area defined
      //  by mouseX and mouseY
```

Result

EDIT ON CODEPEN

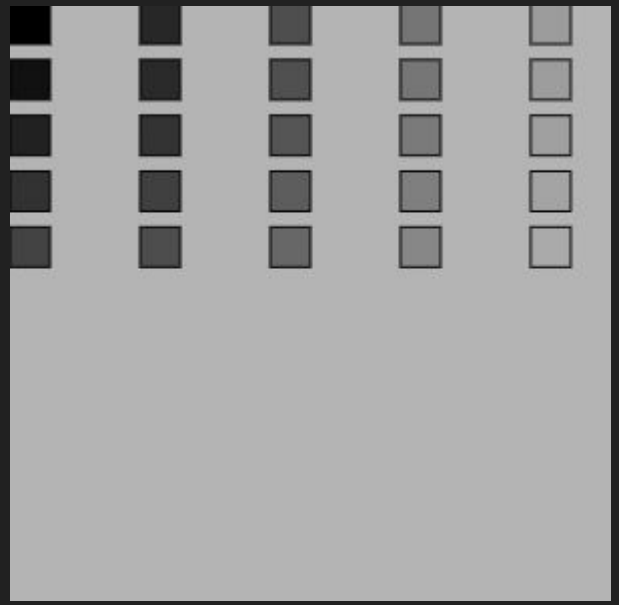


Resources | 1x | 0.5x | 0.25x | Rerun

CityU canvas

exercise

In-class exercise 2



Use the previous `map()` example as your skeleton code; and improve the interaction by using the `lerp()` function as demonstrated in the Natural Motion example.

In addition, each square should have its filled brightness (color) determined by its distance from the origin (0,0). Hint: use both `dist()` and `map()`.