Images

image processing

Chapter 7, Examples 7-1, 7-2, 7-3, 7-4, 7-5, 7-6, Robot 5 (p. 117-118)
Image Processing

- Manipulating pixels of an image (or sequence of images)
Eighties Image Processing
- http://youtu.be/8t9ZfCYzbm8?t=5m
Chroma Key Compositing (“green screen”)

Credit: www.aimersoft.com
LUT (Lookup Table) Colour Grading (e.g. Instagram Filters)
Image Processing Algorithm

for each $x, y$ location in the image:
  get the pixel colour
  set the pixel colour to the "new" colour
Image Processing Algorithm

for each x, y location in the image:
get the pixel colour
set the pixel colour to the "new" colour

position 0, 0
Image Processing Algorithm

for each x, y location in the image:

get the pixel colour

set the pixel colour to the "new" colour

position 0, 0

(52, 75, 116)
Image Processing Algorithm

for each x, y location in the image:
  get the pixel colour
  set the pixel colour to the "new" colour

position 0, 0
(128, 128, 128)
for each x, y location in the image:
  get the pixel colour
  set the pixel colour to the "new" colour

position 1, 0
Image Processing Algorithm

1. for each x, y location in the image:
   get the pixel colour
   set the pixel colour to the "new" colour
Traversing through pixels

position 0, 0
(52, 75, 116)
Traversing through pixels

position 1, 0
(40, 62, 109)
Traversing through pixels

position 2, 0
(42, 63, 110)
Traversing through pixels

position 249, 0
(120, 169, 137)
Traversing through pixels

position 0, 1
(25, 49, 85)
Traversing through pixels

position 0, 1
(25, 49, 85)
Traversing through pixels

position 249, 1
(93, 130, 112)
Traversing through pixels

position 0, 2
(35, 57, 94)
Traversing through pixels

position 249, 2

(74, 126, 78)
Traversing through pixels

After a while...
(or eventually)
Traversing through pixels

position 249, 249
(166, 143,149)
How can we implement this with code?

// traverse through pixels

```
0 1 2 3 4
0 0 1 2 3 4
1 5 6 7 8 9
2 10 11 12 13 14
3 15 16 17 18 19
4 20 21 22 23 24
```

Credit: LP p.262
(Review) Nested Loop from Lecture 12

```javascript
for (let y = 0; y < height; y += 10) {
    for (let x = 0; x < width; x += 10) {
        rect(x, y, 10, 10);
    }
}
```
(Review) Inner loop - left to right

```javascript
for (let y = 0; y < height; y += 10) {
    for (let x = 0; x < width; x += 10) {
        rect(x, y, 10, 10);
    }
}
```
(Review) Outer loop - top to bottom

```javascript
for (let y = 0; y < height; y += 10) {
    for (let x = 0; x < width; x += 10) {
        rect(x, y, 10, 10);
    }
}
```
Use nested loop

```
for (let y = 0; y < 5; y++) {
    for (let x = 0; x < 5; x++) {

    }
}
```

Credit: LP p.262
Image Processing Algorithm

1. for each x,y location in the image:
   - get the pixel colour
   - set the pixel colour to the "new" colour
Image Processing Algorithm

for each x,y location in the image:

2. get the pixel colour

set the pixel colour to the "new" colour
Use nested loop

for (let y = 0; y < 5; y++) {
    for (let x = 0; x < 5; x++) {
        // get the pixel colour
    }
}

Credit: LP p.262
Pixels are stored in a single array

```javascript
for (let y = 0; y < 5; y++) {
    for (let x = 0; x < 5; x++) {
        // get the pixel colour
    }
}
```

Credit: LP p.262
Pixels are stored in a single array

```
for (let y = 0; y < 5; y++) {
    for (let x = 0; x < 5; x++) {
        // get the pixel colour
        pixels[?]
    }
}
```

Credit: LP p.262
That is to say...

```javascript
for (let y = 0; y < 5; y++) {
    for (let x = 0; x < 5; x++) {
        // get the pixel colour at (0, 0)
        pixels[0]
    }
}
```

Credit: LP p.262
That is to say...

```javascript
for (let y = 0; y < 5; y++) {
  for (let x = 0; x < 5; x++) {
    // get the pixel colour at (1, 0)
    pixels[1]
  }
}
```

Credit: LP p.262
That is to say...

```javascript
for (let y = 0; y < 5; y++) {
    for (let x = 0; x < 5; x++) {
        // get the pixel colour at (2, 0)
        pixels[2]
    }
}
```

Credit: LP p.262
That is to say...

```javascript
for (let y = 0; y < 5; y++) {
    for (let x = 0; x < 5; x++) {
        // get the pixel colour at (3, 0)
        pixels[3]
    }
}
```

Credit: LP p.262
That is to say...

```javascript
for (let y = 0; y < 5; y++) {
  for (let x = 0; x < 5; x++) {
    // get the pixel colour at (4, 0)
    pixels[4]
  }
}
```

Credit: LP p.262
That is to say...

```javascript
for (let y = 0; y < 5; y++) {
    for (let x = 0; x < 5; x++) {
        // get the pixel colour at (0, 1)
        pixels[5]
    }
}
```

Current Pixel

Current Index

Credit: LP p.262
That is to say...

```javascript
for (let y = 0; y < 5; y++) {
    for (let x = 0; x < 5; x++) {
        // get the pixel colour at (1, 1)
        pixels[6]
    }
}
```

Credit: LP p.262
That is to say...

```javascript
for (let y = 0; y < 5; y++) {
    for (let x = 0; x < 5; x++) {
        // get the pixel colour at (3, 2)
        pixels[?]}
    }
}
```

Credit: LP p.262
That is to say...

```javascript
for (let y = 0; y < 5; y++) {
  for (let x = 0; x < 5; x++) {
    // get the pixel colour at (2, 3)
    pixels[?]
  }
}
```

need to calculate index value based on x, y

Credit: LP p.262
Mapping Formula

let \textbf{index} = x + y \times w; // w is image width

Pixel 13 is in Column 3, row 2.

\begin{align*}
  x + y \times w \\
  \downarrow \\
  3 + 2 \times 5 \\
  3 + 10 \\
  13
\end{align*}

Credit: LP p.263
Black and White (Binary) Images
```javascript
for (let y = 0; y < h; y++) {
    for (let x = 0; x < w; x++) {
        let index = x + y * w;
        if (pixels[index]) {
            stroke(255);
        } else {
            stroke(0);
        }
        point(x, y);
    }
}

let w = 64;
let h = 64;
let pixels = [true, false, true, false, true, true, true, ...]
```

https://editor.p5js.org/cs105/sketches/aMPjW4SUH
Grayscale Images
noStroke();
    for (let y = 0; y < h; y++) {
        for (let x = 0; x < w; x++) {
            let index = x + y * w; // indexing
            let gray = pixels[index];
            stroke(gray)
            point(x, y);
        }
    }

let w = 64;
let h = 64;
let pixels = [162, 148, 144, 145, 146, 152, ... ]

https://editor.p5js.org/cs105.sketches/OrJaDJOYY
RGB Image
RGB Image
RGB Image Data Stored in One Array

- Each pixel has 3 numbers (Red, Green, Blue ... RGB)
- Every three elements is the colour for a pixel
  - indices 0, 1, 2 are the RGB colour for image pixel (0,0)
  - indices 3, 4, 5 are the RGB colour for image pixel (1,0)
  - etc.
- Assume image is stored row-by-row
- If image is $w$ pixels wide, then index of red for image pixel $x, y$ is:
  $\text{index} = (x + y \times w) \times 3$

```
0 1 2 3 4 5 6 7 ...
```
function setup() {
    createCanvas(w, h);
    noStroke();
    for (let y = 0; y < h; y++) {
        for (let x = 0; x < w; x++) {
            let i = (x + y * w) * 3; // indexing
            stroke(pixels[i], pixels[i+1], pixels[i+2]);
            point(x, y);
        }
    }
}

// image size
let w = 48;
let h = 48;

// RGB image stored like this: r, g, b, r, g, b, ...
let pixels = [229,97,56,221,81,26,220,84,19,220,91,16,219,102,14,220,114,14,221,127,14,219, ...]

https://editor.p5js.org/cs105/sketches/qi8sbbikv
RGB Image “in p5”
How p5 Stores Image Data in One Array

- Each pixel has 4 numbers (Red, Green, Blue, Alpha ... RGBA)
- Every four elements is the colour for a pixel
  - indices 0, 1, 2, 3 are the RGBA colour for image pixel (0,0)
  - indices 4, 5, 6, 7 are the RGBA colour for image pixel (1,0)
  - etc.
- Image is stored row-by-row
- If image is $w$ pixels wide, then index of red for image pixel $x, y$ is:
  $$\text{index} = (x + y \times w) \times 4$$

https://p5js.org/reference/#/p5/pixels
loadPixels()

- We’ve been drawing using toy examples of pixels array prepared in advance for you to get a sense of what pixels array looks like and how it works
- Use loadPixels() to load pixels array

https://p5js.org/reference/#/p5/loadPixels
// load the pixels array
img.loadPixels();

// index into image pixel array
let i = (mouseX + mouseY * img.width) * 4;

// extract red, green, blue, alpha
let r = img.pixels[i];
let g = img.pixels[i + 1];
let b = img.pixels[i + 2];
let a = img.pixels[i + 3];

fill(r, g, b);

https://editor.p5js.org/cs105/sketches/S2KyZaYRh
Image Processing Algorithm

for each x, y location in the image:

2. get the pixel colour

set the pixel colour to the "new" colour
Image Processing Algorithm

for each x,y location in the image:
get the pixel colour

set the pixel colour to the "new" colour
for (let x = 0; x < img.width; x++) {
    for (let y = 0; y < img.height; y++) {
        // extract red, green, blue, alpha from pixel
        let i = (x + y * img.width) * 4; // index into pixel array
        let r = img.pixels[i];
        let g = img.pixels[i + 1];
        let b = img.pixels[i + 2];
        let a = img.pixels[i + 3];
        // process pixel here ... 
        // write red, blue, green, alpha back to pixel
        img.pixels[i] = r;
        img.pixels[i + 1] = g;
        img.pixels[i + 2] = b;
        img.pixels[i + 3] = a;
    }
}

for each x, y in the image:
```javascript
for (let x = 0; x < img.width; x++) {
    for (let y = 0; y < img.height; y++) {
        let i = (x + y * img.width) * 4; // index into pixel array

        // extract red, green, blue, alpha from pixel
        let r = img.pixels[i];
        let g = img.pixels[i + 1];
        let b = img.pixels[i + 2];
        let a = img.pixels[i + 3];

        // process pixel here ...

        // write red, blue, green, alpha back to pixel
        img.pixels[i] = r;
        img.pixels[i + 1] = g;
        img.pixels[i + 2] = b;
        img.pixels[i + 3] = a;
    }
}
```
for (let x = 0; x < img.width; x++) {
    for (let y = 0; y < img.height; y++) {
        let i = (x + y * img.width) * 4; // index into pixel array

        // extract red, green, blue, alpha from pixel
        let r = img.pixels[i];
        let g = img.pixels[i + 1];
        let b = img.pixels[i + 2];
        let a = img.pixels[i + 3];

        // process pixel here ...

        // write red, blue, green, alpha back to pixel
        img.pixels[i] = r;
        img.pixels[i + 1] = g;
        img.pixels[i + 2] = b;
        img.pixels[i + 3] = a;
    }
}
for (let x = 0; x < img.width; x++) {
    for (let y = 0; y < img.height; y++) {

        let i = (x + y * img.width) * 4; // index into pixel array

        // extract red, green, blue, alpha from pixel
        let r = img.pixels[i];
        let g = img.pixels[i + 1];
        let b = img.pixels[i + 2];
        let a = img.pixels[i + 3];

        // process pixel here ...

        // write red, blue, green, alpha back to pixel
        img.pixels[i] = r;
        img.pixels[i + 1] = g;
        img.pixels[i + 2] = b;
        img.pixels[i + 3] = a;
    }
}
let contrast = 3;
r = r * contrast;
g = g * contrast;
b = b * contrast;

https://editor.p5js.org/sanghosuh.sketches/a99lW1Iqg
(gradient)

```javascript
let f = map(y, 0, img.height, 1, 0);
r = r * f;
g = g * f;
```

https://editor.p5js.org/sanghosuh/sketches/XMSC0wmFE
(simple motion blur)

// choose random pixel nearby in same row
let x2 = x + floor(random(-5, 1));
// make sure it doesn't go out of bounds
x2 = constrain(x2, 0, img.width - 1);
// get the index for this nearby pixel
let j = (x2 + y * img.width) * 4;
// copy the value
r = img.pixels[j];
g = img.pixels[j + 1];
b = img.pixels[j + 2];
a = img.pixels[j + 3];

https://editor.p5js.org/sanghosuh.sketches/IsdPB-cy7
(threshold)

// get % brightness from RGB colour
let f = brightness(color(r, g, b));
if (f > 50) {
    // make pixel white
    r = 255;
    b = 255;
    g = 255;
} else {
    // make pixel black
    r = 0;
    b = 0;
    g = 0;
}

https://editor.p5js.org/sanghosuh/sketches/aDcMU6Yyg
for (let x = 0; x < width; x++) {
    for (let y = 0; y < height; y++) {

        let i = (x + y * width) * 4; // index into pixel array

        // extract red, green, blue, alpha from pixel
        let r = pixels[i];
        let g = pixels[i + 1];
        let b = pixels[i + 2];
        let a = pixels[i + 3];

        // process pixel here ...

        // write red, blue, green, alpha back to pixel
        pixels[i] = r;
        pixels[i + 1] = g;
        pixels[i + 2] = b;
        pixels[i + 3] = a;
    }
}
process-image vs process-canvas

https://editor.p5js.org/sanghosuh/sketches/Hnvbk-pKT

https://editor.p5js.org/sanghosuh/sketches/4iIENWBo
Pixel Density and "Retina" displays

- By default, canvas pixels are rendered using display pixel density
- A “Retina Display” has a pixelDensity of 2:
  - i.e. 2 x 2 display pixels for each canvas pixel
- To use canvas width and height for indexing into the pixels array, set the pixelDensity to 1 like this:
  ```javascript
  pixelDensity(1);
  ```

let threshold = map(mouseY, 0, height, 0, 100);
if (f > threshold) {
    // make pixel white
} else {
    // make pixel black
}
// how far is pixel from mouse?
let d = dist(mouseX, mouseY, x, y);
// make it so farther pixels remove all colour
let f = map(d, 0, 100, 1, 0);
r = r * f;
g = g * f;
b = b * f;

https://editor.p5js.org/sanghosuh/sketches/oqHgZMypr
You can image process anything that you draw on the canvas, not just images.

https://editor.p5js.org/cs105/sketches/P6oKufj3T
11.3: The Pixel Array - p5.js Tutorial

Image Processing with Pixels
https://www.youtube.com/watch?v=nMUMZ5YRxHI
https://evaluate.uwaterloo.ca/