Image Manipulation: Pixel Traversal

Elements of Graphics
CS324e
Fall 2017
Student Presentation
Digital Images

- Bits are binary (0 or 1)
- Pixels are composed of bits
  - Bits-per-pixel determine the range of color
- Images are composed of pixels
Image Buffers

- Screen pixel data is stored in an array
- This array (or image buffer) allows us to access per-pixel information

How the pixels look:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td></td>
</tr>
</tbody>
</table>

How the pixels are stored:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th></th>
</tr>
</thead>
</table>
Images in Processing

- Image buffers are stored in the `PImage` data type
- `PImage` allows for loading and displaying image data
- Some image manipulation:
  - Size
  - Position
  - Opacity
  - Tint
- To display: `image(PImage img, float x, float y, float width, float height)`
Loading and Displaying Images

PImage img;

void setup() {
    size(100, 100);
    img = loadImage("foo.png");
}

void draw() {
    image(img, 0, 0);
}
Fitting Processing Window to Image Size

```java
void setup() {
    surface.setResizable(true);
    img = loadImage("foo.png");
    surface.setSize(img.width, img.height);
}
```
Changing Individual Pixels

- `loadPixels()` and `updatePixels()` should be called before and after pixel manipulation respectively.

- `PImage.pixels` array stores each pixel as a `color`.

- Access the color of the pixel at index in `PImage img`:
  - `color c = img.pixels[index];`
Consider…

❖ How should we access every pixel in an image?
Traversing an Image Buffer

//access img’s pixels

img.loadPixels();

for (int x = 0; x < img.width; x++) {
    for (int y = 0; y < img.height; y++) {
        //access pixel at index and set c to its value
        int index = x + y*img.width;
        color c = img.pixels[index];
    }
}

//update any modifications to img’s pixels

img.updatePixels();
Tint

- `tint()` modifies the color of the displayed images
- `noTint()` disables `tint()` modifications

```plaintext
void draw() {
    tint(0, 153, 204);
    image(img, 0, 0);
    noTint();
    image(img, 50, 50);
}
```
Opacity

- Parameters for `tint()`’s color space is determined by `colorMode()`
- `tint()` can also modify image alpha channels
- What is the difference between these calls?
  ```
  tint(255, 110);
  tint(255, 255, 255, 110);
  ```
Hands-on: Creating Tint

❖ Today’s activity:

1. Re-create Processing’s `tint` functionality using your own method