

# Green Screen Web Page

Convert Image to Grayscale


# Changing an Uploaded Image

- In previous lesson, you saw how to modify pixels in an image, to make different parts of the image different colors
- You saw how to combine images to create a Green Screen effect
- We'll use the same concepts so you can create interactive web pages with the SimpleImage library

# Grayscale

- Start with page that uploads and displays an image
- Add a button to display grayscale version

**Convert to Grayscale**

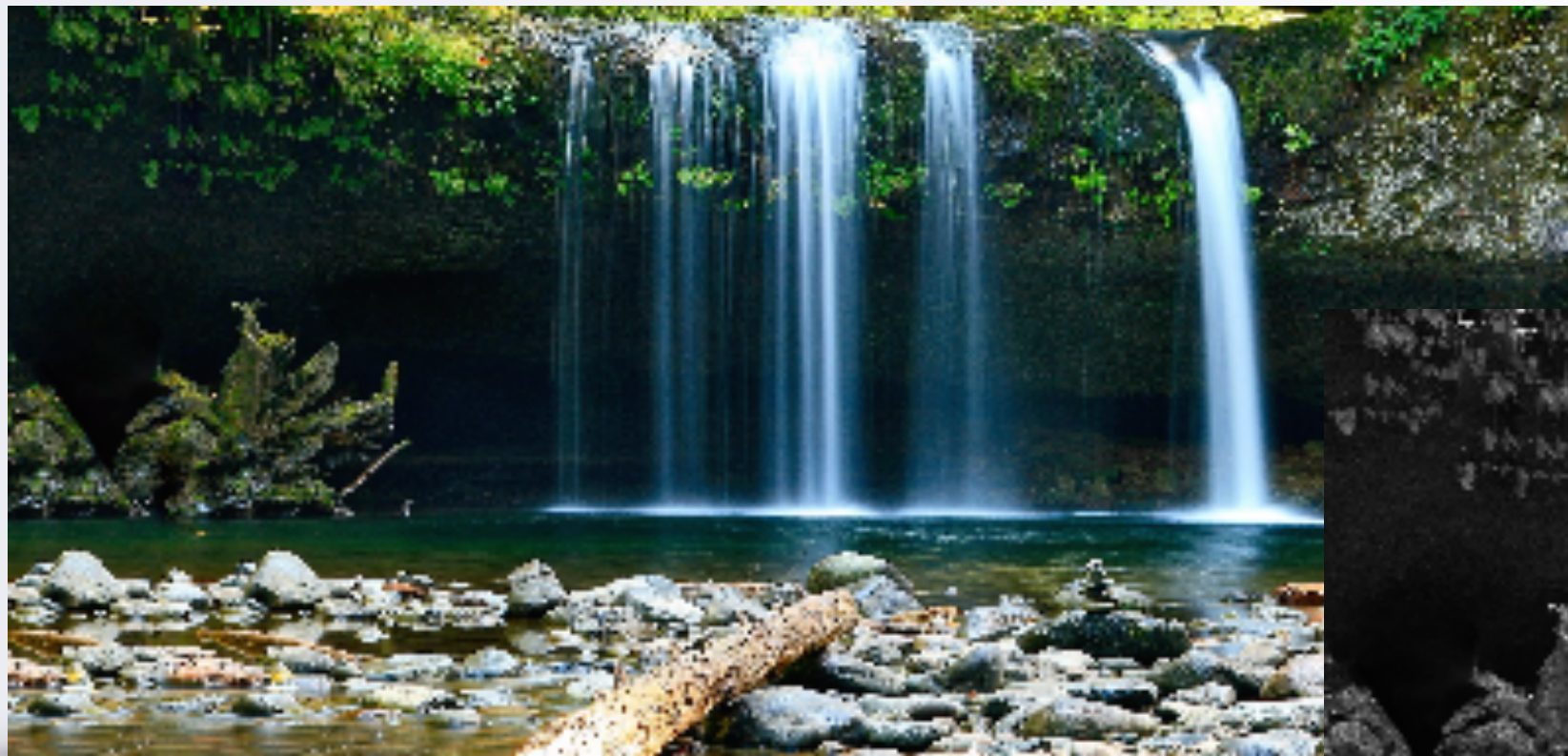


No file chosen



# Grayscale

- Start with page that uploads and displays an image
- Add a button to display grayscale version

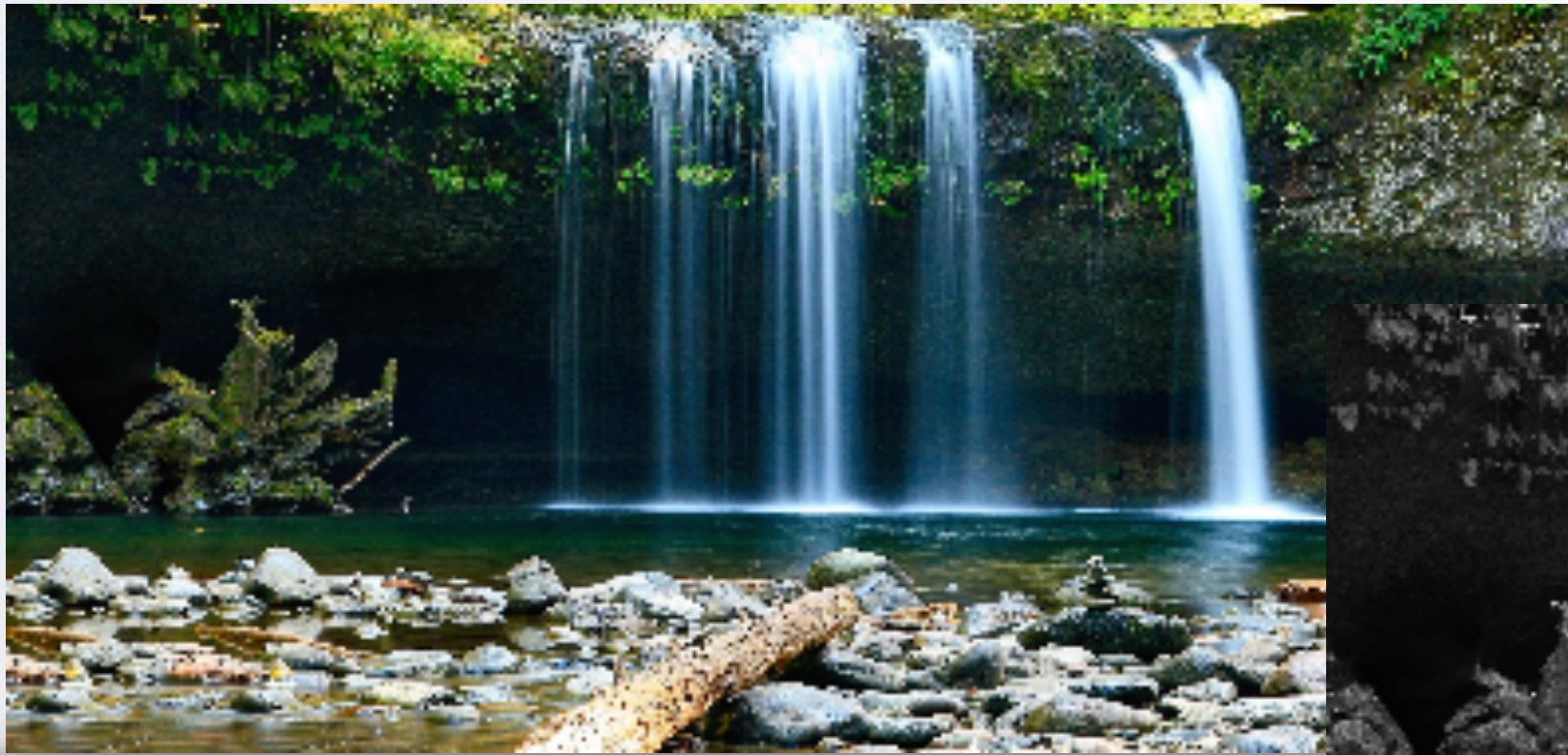


# Grayscale

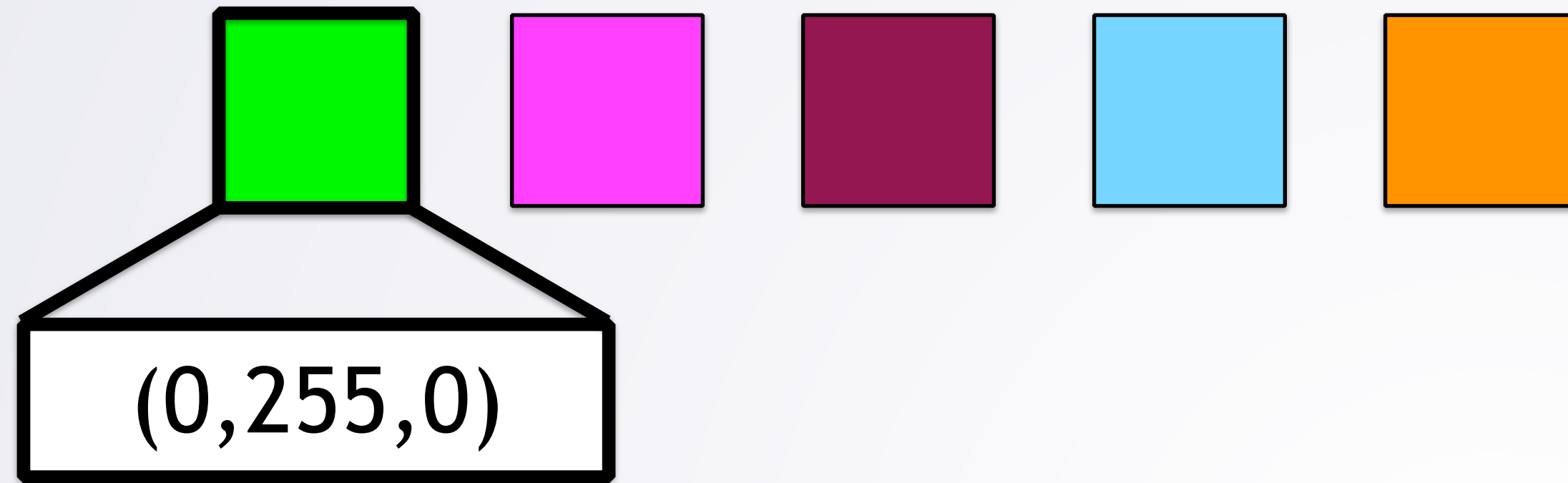
- Start with page that uploads and displays an image
- Add a button to display grayscale version
  - Access uploaded image in `makeGray()`
  - We'll need a new concept: *global variables*
- Access global variable in all functions
  - `makegray()` and `upload()`
  - Set variable and access variable



# Grayscale Algorithm?

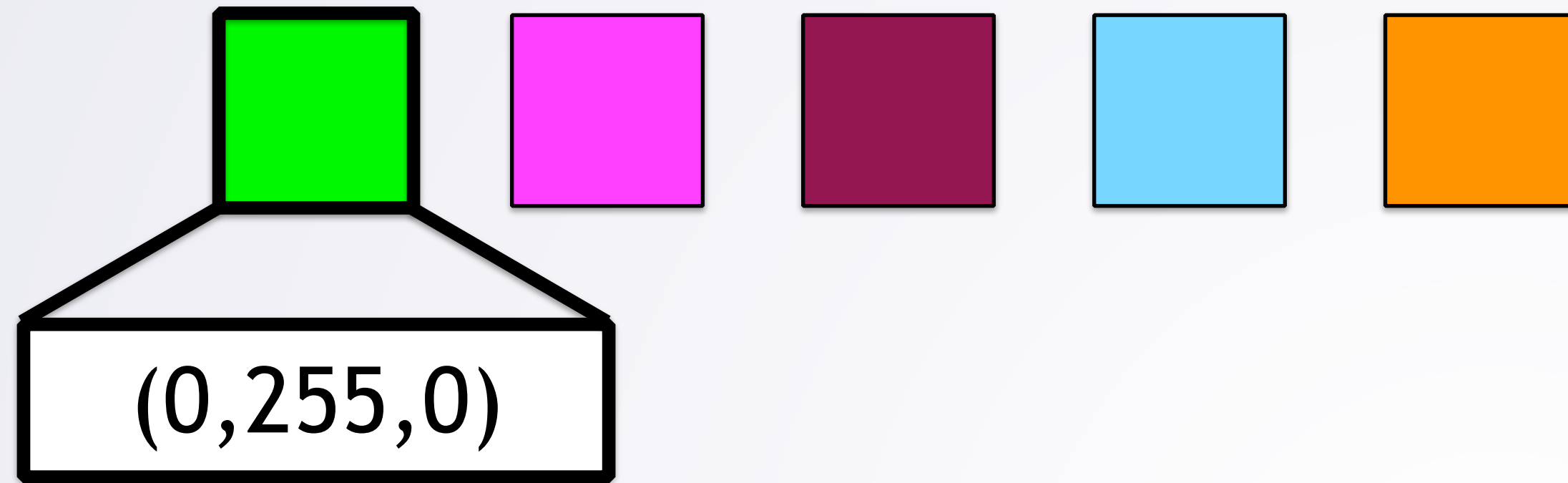


# Grayscale by Hand

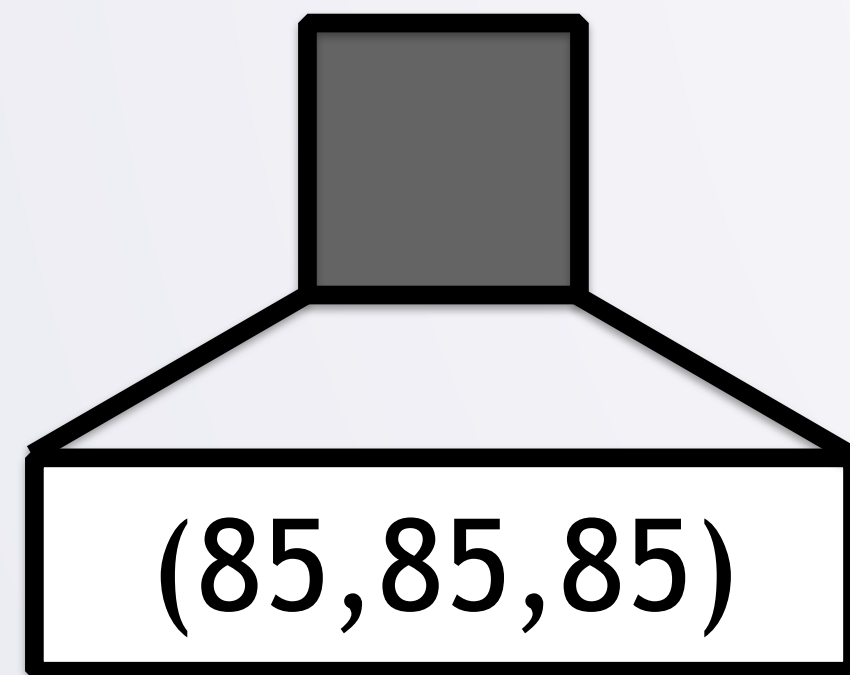


$$\text{Average} = (0 + 255 + 0) / 3 = 85$$

# Grayscale by Hand

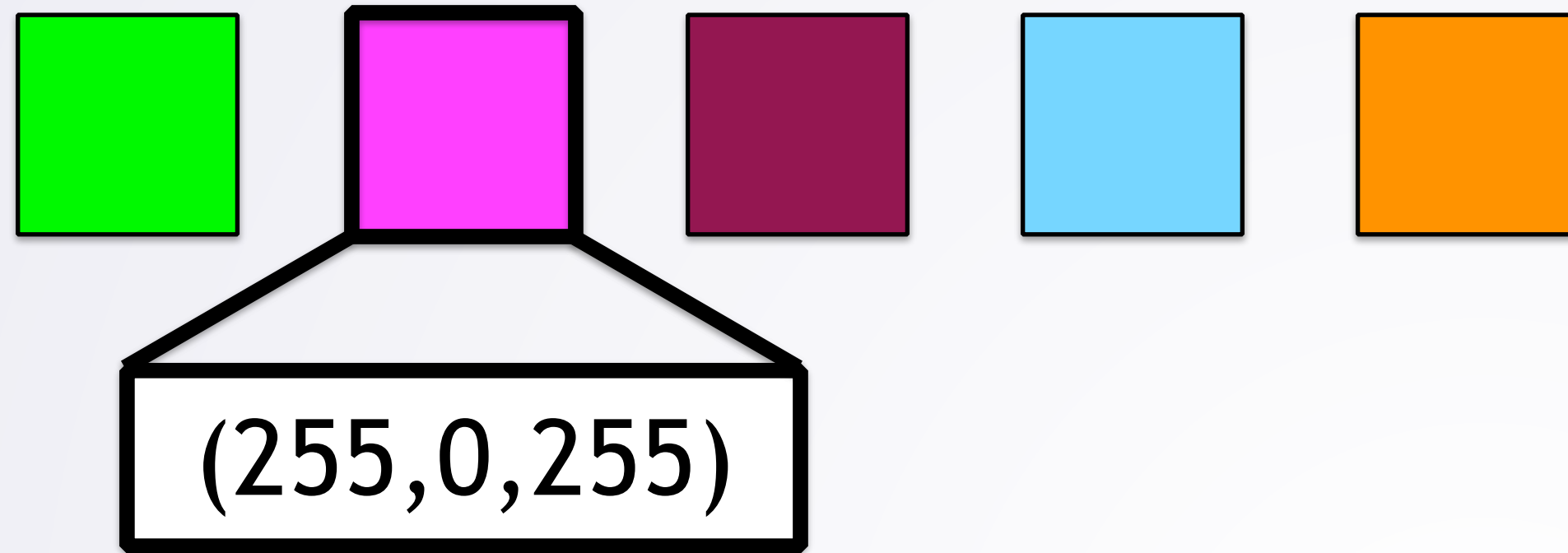


$$\text{Average} = (0 + 255 + 0) / 3 = 85$$

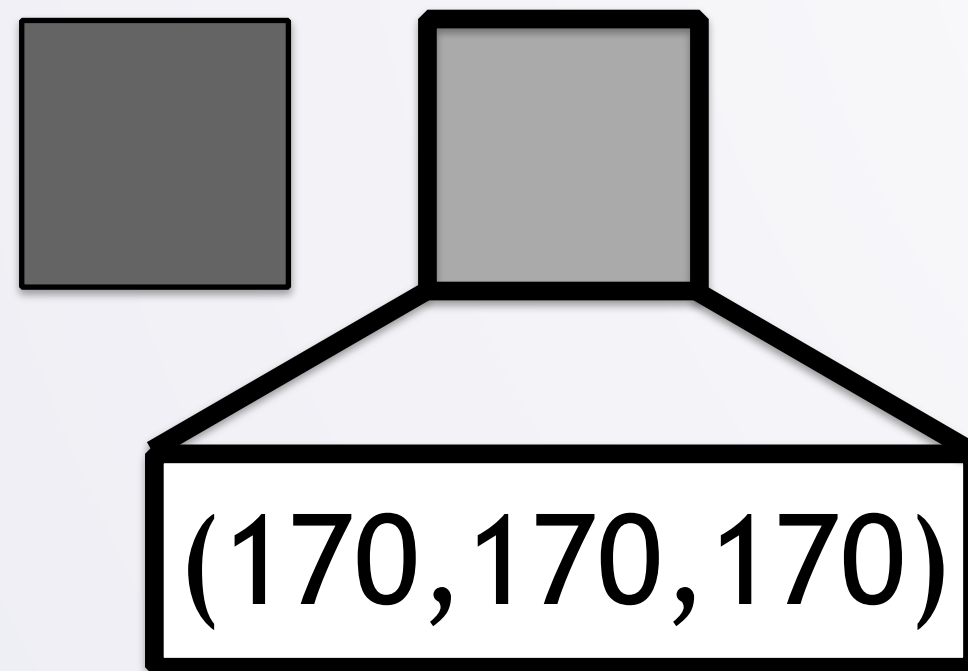




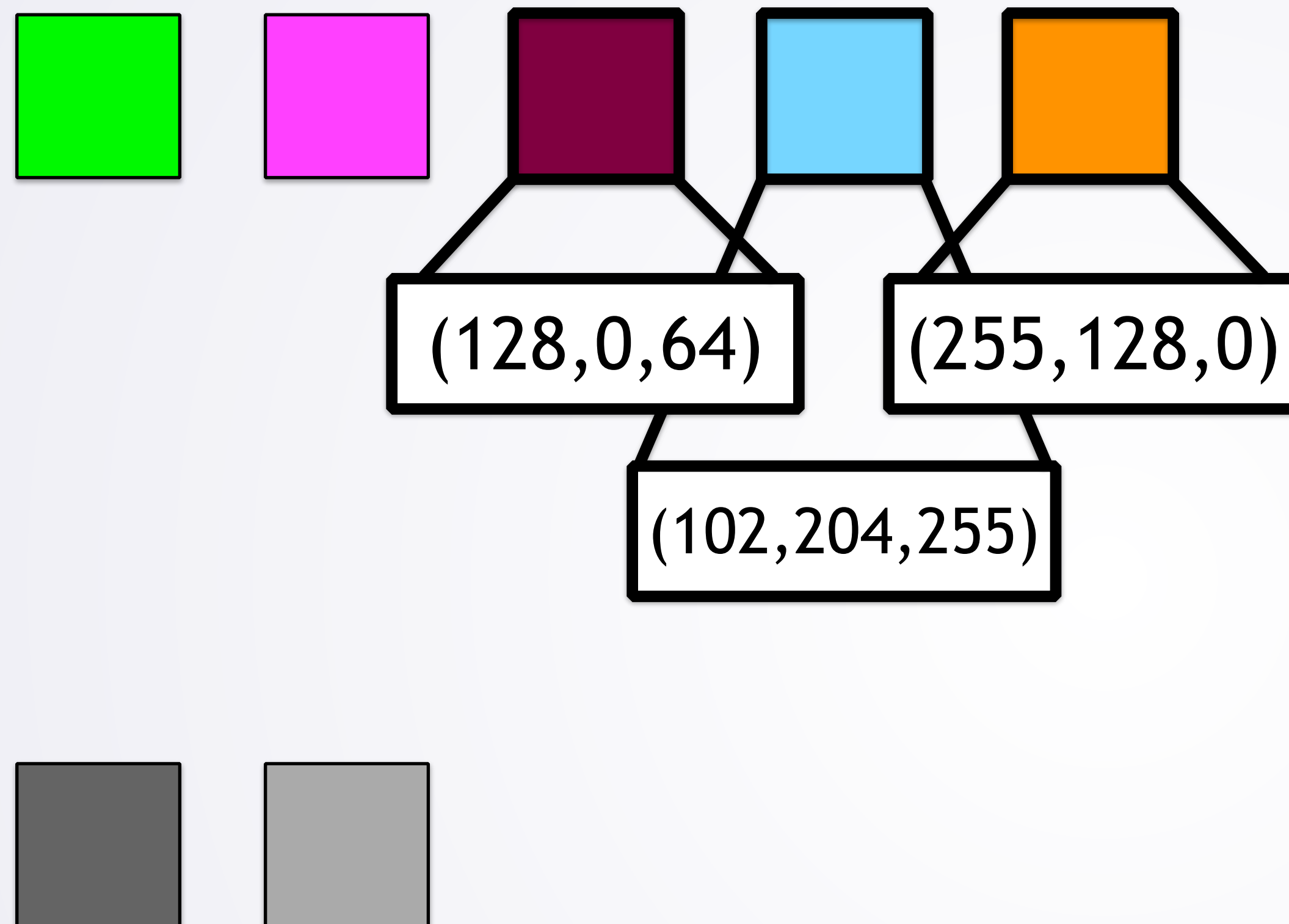
# Grayscale by Hand



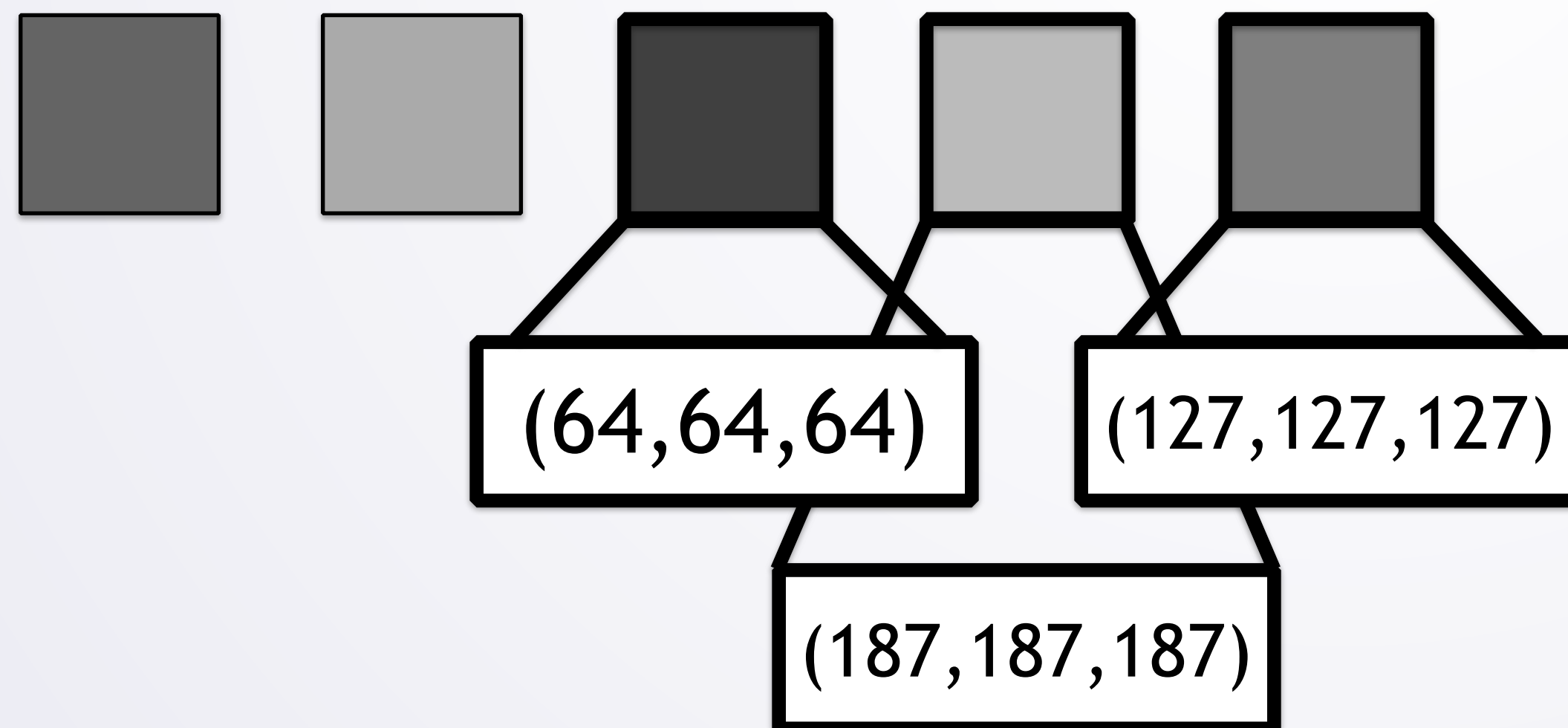
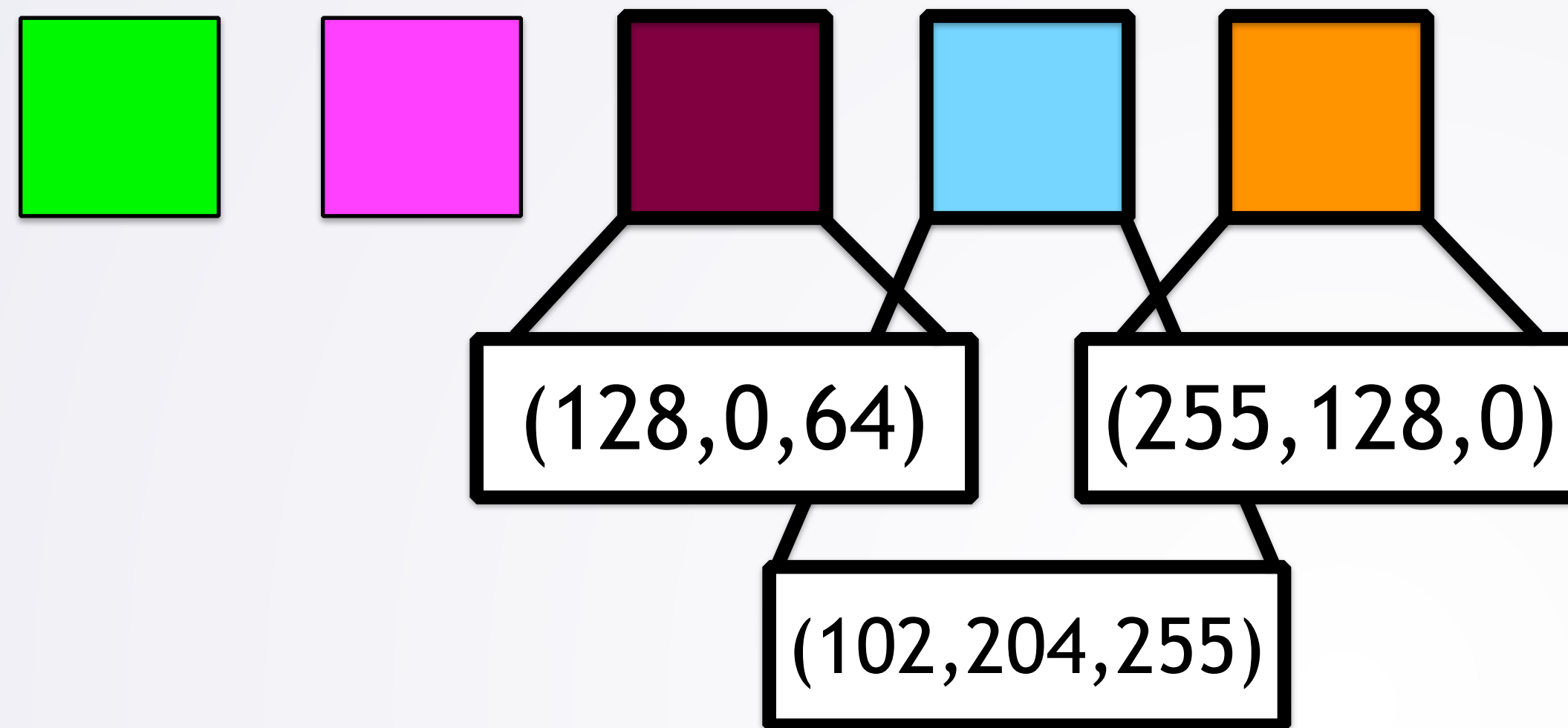
$$\text{Average} = (255 + 0 + 255) / 3 = 170$$



# Grayscale by Hand



# Grayscale by Hand





# Grayscale Algorithm

- 1 Start with the image you want
- 2 For each pixel in image
  1. Get pixel's red, green, and blue values
  2. Calculate the average value
  3. Set each of red, green, and blue values to average value
- 3 Display the final image

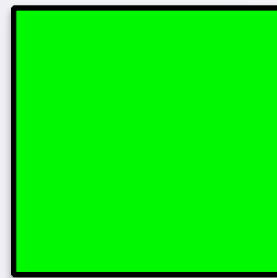
# Grayscale Algorithm

① Start with the image you want

② For each pixel in image

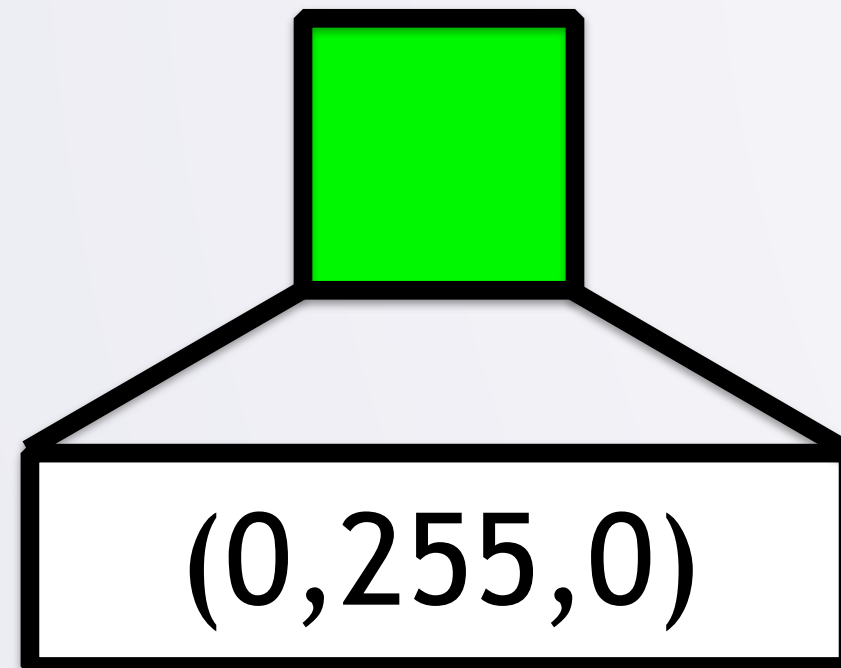
1. Get pixel's red, green, and blue values
2. Calculate the average value
3. Set each of red, green, and blue values to average value

③ Display the final image



# Grayscale Algorithm

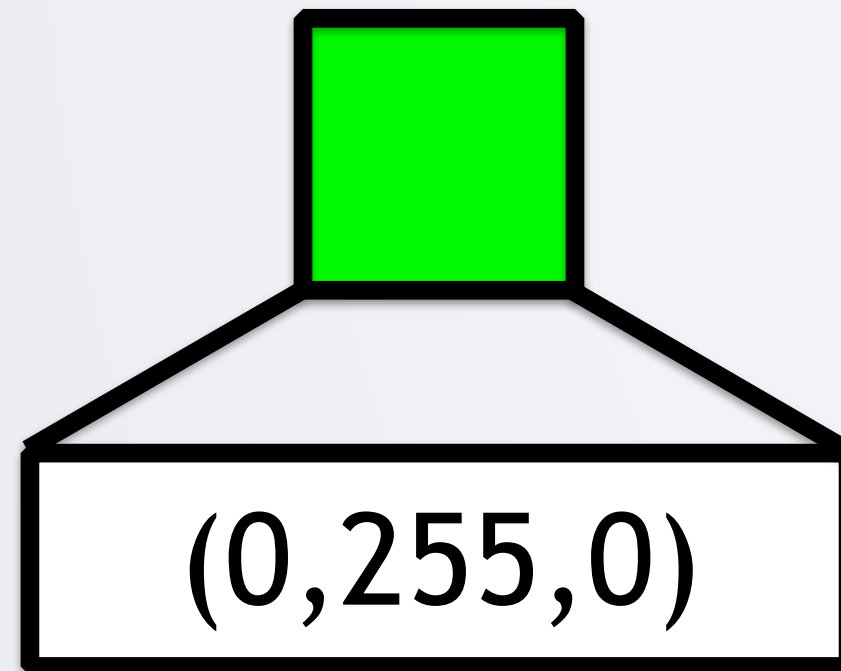
- ① Start with the image you want
- ② For each pixel in image
  1. Get pixel's red, green, and blue values
  2. Calculate the average value
  3. Set each of red, green, and blue values to average value
- ③ Display the final image





# Grayscale Algorithm

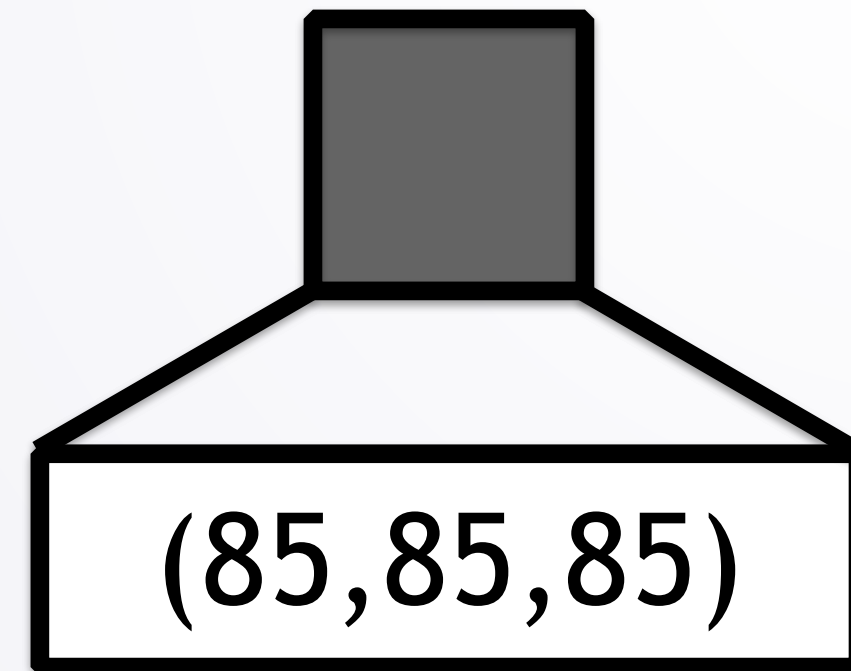
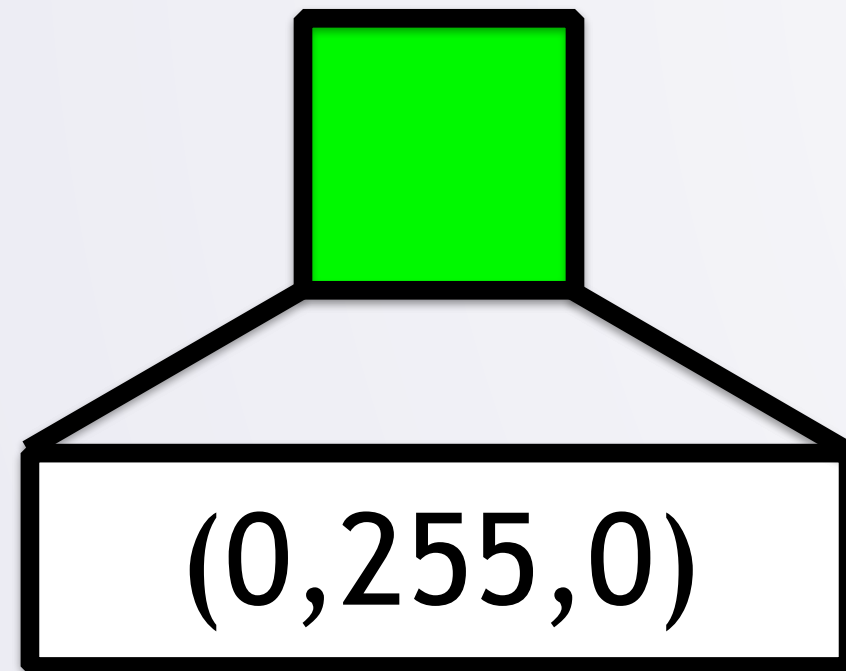
- ① Start with the image you want
- ② For each pixel in image
  1. Get pixel's red, green, and blue values
  2. Calculate the average value
  3. Set each of red, green, and blue values to average value
- ③ Display the final image



$$\text{Average} = (0 + 255 + 0) / 3 = 85$$

# Grayscale Algorithm

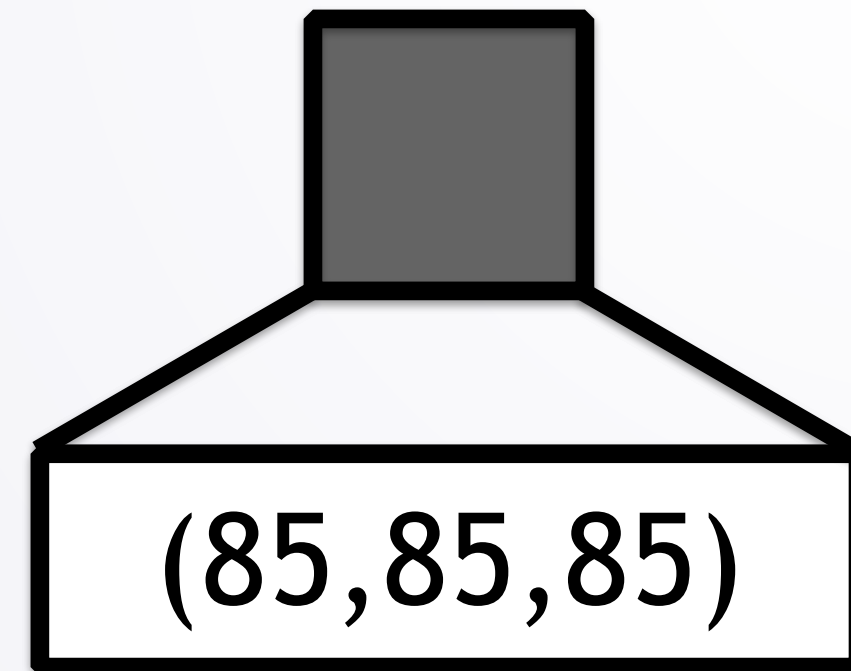
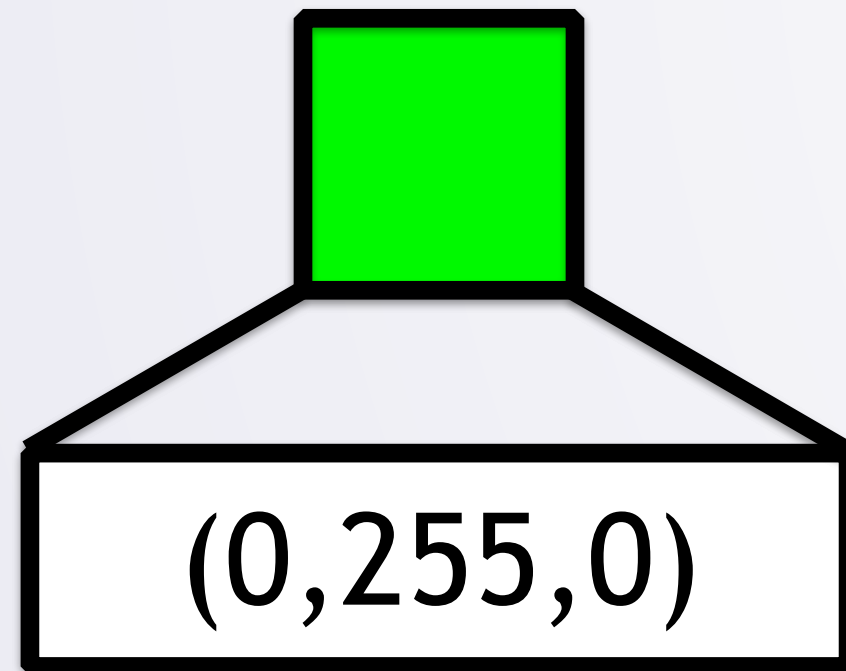
- 1 Start with the image you want
- 2 For each pixel in image
  1. Get pixel's red, green, and blue values
  2. Calculate the average value
  3. Set each of red, green, and blue values to average value
- 3 Display the final image



$$\text{Average} = (0 + 255 + 0) / 3 = 85$$

# Grayscale Algorithm

- ① Start with the image you want
- ② For each pixel in image
  1. Get pixel's red, green, and blue values
  2. Calculate the average value
  3. Set each of red, green, and blue values to average value
- ③ Display the final image



$$\text{Average} = (0 + 255 + 0) / 3 = 85$$



# Translate to Code

**Convert to Grayscale**

No file chosen

`<input type="button" value="Make  
Grayscale" onclick="makeGray()" >`

# makeGray() Function

```
function makeGray() {  
    for (var pixel of image.values()) {  
        var avg = (pixel.getRed()+pixel.getGreen()  
+pixel.getBlue())/3;  
        pixel.setRed(avg);  
        pixel.setGreen(avg);  
        pixel.setBlue(avg);  
    }  
    var imgcanvas = document.getElementById("can");  
    image.drawTo(imgcanvas);  
}
```

# Where Is Image?

```
function upload() {  
    [get file input]  
    var image = new SimpleImage(fileinput);  
    image.drawTo(imgcanvas);  
}
```

---

```
function makeGray() {  
    for (var pixel of image.values()) { ???  
        [process pixels]  
    }  
    var imgcanvas = document.getElementById("can");  
    image.drawTo(imgcanvas);  
}
```



# Where Is Image?

upload()

`var` image

makeGray()

image.~~X~~values()

# Where Is Image?

global variable:

`var image`

`upload()`

`image`

`makeGray()`

`image.values()`

# Global Variables

```
var image;
```

→ defined outside of all functions

```
function upload() {  
    [code not shown]  
    image = new SimpleImage(fileinput);  
}
```

```
function makeGray() {  
    for (var pixel of image.values()) {  
        [process pixels]  
    }  
    var imgcanvas = document.getElementById("can");  
    image.drawTo(imgcanvas);  
}
```



# Global Variables

```
var image;
```

```
function upload() {  
    [code not shown]  
    image = new SimpleImage(fileinput);  
}
```

**no *var* inside functions**

```
function makeGray() {  
    for (var pixel of image.values()) {  
        [process pixels]  
    }  
    var imgcanvas = document.getElementById("can");  
    image.drawTo(imgcanvas);  
}
```

# Many Global Variables

- Suppose we want to display the original image again, without reloading
  - Can create a new image rather than modifying the global variable image
  - Store regular and grayscale in global variables
  - You can try this out
- In general: take care with global variables