

CSc 110

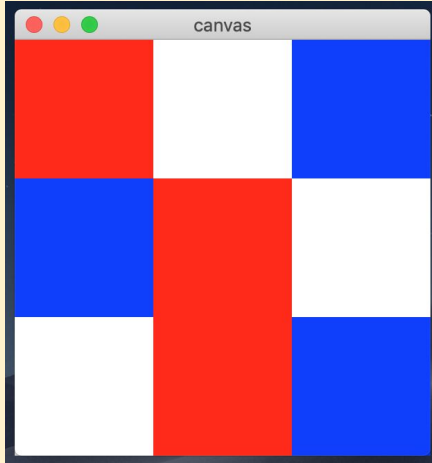
PPM

Benjamin Dicken



Display
the grid

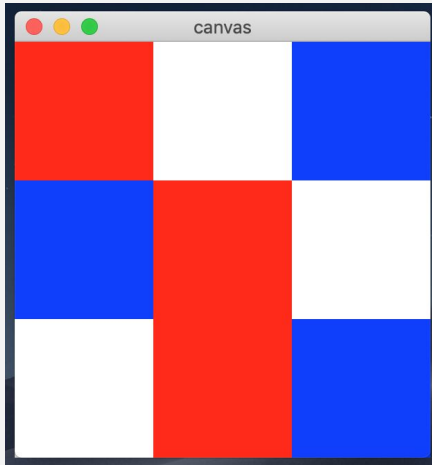
```
colors = [ ['red', 'white', 'blue'],  
           ['blue', 'red', 'white'],  
           ['white', 'red', 'blue']]
```



```
gui = graphics(300, 300, 'canvas')
```

Complete the code

Display
the grid



```
colors = [ ['red', 'white', 'blue'],  
           ['blue', 'red', 'white'],  
           ['white', 'red', 'blue']]
```

```
gui = graphics(300, 300, 'canvas')
```

```
x = 0
```

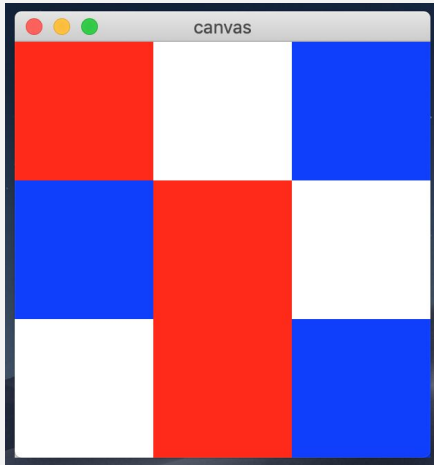
```
y = 0
```

```
for i in colors:
```

```
    for j in i:
```

```
        gui.rectangle(x, y, 100, 100, j)
```

Display the grid



```
colors = [ ['red', 'white', 'blue'],  
           ['blue', 'red', 'white'],  
           ['white', 'red', 'blue']]
```

```
gui = graphics(300, 300, 'canvas')
```

```
x = 0
```

```
y = 0
```

```
for i in colors:
```

```
    x = 0
```

```
    for j in i:
```

```
        gui.rectangle(x, y, 100, 100, j)
```

```
        x += 100
```

```
    y += 100
```

```
colors = [ ['red', 'white', 'blue'],  
            ['blue', 'red', 'white'],  
            ['white', 'red', 'blue']]
```

```
gui = graphics(300, 300, 'canvas')
```

```
for i in range(len(colors)):  
    for j in range(len(colors[0])):  
        gui.rectangle(j * 100, i * 100, 100, 100, colors[i][j])
```

Storing image data

- Discuss
 - How would you read a PPM file in python?
 - What data structure(s) would you use to store PPM image data so that you could access all of the rows, columns, and RGB values?

P3

4 2

255

255 0 0	255 0 0	0 0 255	0 0 255
0 255 0	0 255 0	50 50 50	50 50 50



```
image = [ [[255, 0, 0], [255, 0, 0], [0, 0, 255], [0, 0, 255]],  
          [[0, 255, 0], [0, 255, 0], [50, 50, 50], [50, 50, 50]] ]
```

Read a PPM file

```
image = []  
file_name = input('Enter file name: ')  
ppm_file = open(file_name, 'r')
```

Write the missing code to read in the PPM file

Read a PPM file

```
image = []  
file_name = input('Enter file name: ')  
ppm_file = open(file_name, 'r')  
ppm_file.readline()  
ppm_file.readline()  
ppm_file.readline()
```

?

```
P3  
3 3  
255  
255 255 255 255 255 255 0 100 200  
255 255 255 0 100 200 0 100 200  
0 100 200 0 100 200 255 255 255
```

```
image = []  
file_name = input('Enter file name: ')  
ppm_file = open(file_name, 'r')  
ppm_file.readline()  
ppm_file.readline()  
ppm_file.readline()
```

```
for line in ppm_file:  
    line = line.strip('\n').split()  
    # ?
```

```
P3  
3 3  
255  
255 255 255 255 255 255 0 100 200  
255 255 255 0 100 200 0 100 200  
0 100 200 0 100 200 255 255 255
```

```
image = []  
file_name = input('Enter file name: ')  
ppm_file = open(file_name, 'r')  
ppm_file.readline()  
ppm_file.readline()  
ppm_file.readline()
```

```
for line in ppm_file:  
    line = line.strip('\n').split()  
    row = []  
    # ?
```

```
P3  
3 3  
255  
255 255 255 255 255 255 0 100 200  
255 255 255 0 100 200 0 100 200  
0 100 200 0 100 200 255 255 255
```

```
image = []  
file_name = input('Enter file name: ')  
ppm_file = open(file_name, 'r')  
ppm_file.readline()  
ppm_file.readline()  
ppm_file.readline()
```

```
for line in ppm_file:  
    line = line.strip('\n').split()  
    row = []  
    for i in range(0, len(line), 3):  
        # ?
```

```
P3  
3 3  
255  
255 255 255 255 255 255 0 100 200  
255 255 255 0 100 200 0 100 200  
0 100 200 0 100 200 255 255 255
```

```
image = []  
file_name = input('Enter file name: ')  
ppm_file = open(file_name, 'r')  
ppm_file.readline()  
ppm_file.readline()  
ppm_file.readline()
```

```
for line in ppm_file:  
    line = line.strip('\n').split()  
    row = []  
    for i in range(0, len(line), 3):  
        pixel = line[i:i+3]  
        row.append(pixel)  
    image.append(row)
```

Get the width and height

How would you get the width and height here?



```
image = []  
file_name = input('Enter file name: ')  
ppm_file = open(file_name, 'r')  
ppm_file.readline()  
ppm_file.readline()  
ppm_file.readline()
```

```
for line in ppm_file:  
    line = line.strip('\n').split()  
    row = []  
    for i in range(0, len(line), 3):  
        pixel = line[i:i+3]  
        row.append(pixel)  
    image.append(row)
```

```
P3  
3 3  
255  
255 255 255 255 255 255 200 100 100  
255 255 255 200 100 100 200 100 100  
200 100 100 200 100 100 255 255 255
```

Get the width and height

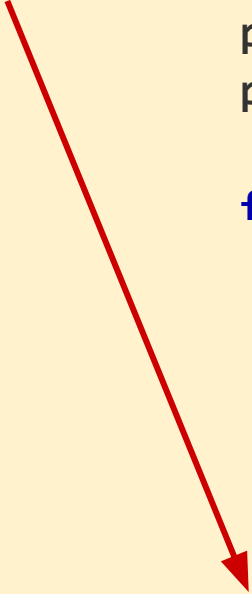
```
w_h = ppm_file.readline()
sp = w_h.split(' ')
width = int(sp[0])
height = int(sp[1])
```

```
image = []
file_name = input('Enter file name: ')
ppm_file = open(file_name, 'r')
ppm_file.readline()
ppm_file.readline()
ppm_file.readline()
```

```
for line in ppm_file:
    line = line.strip('\n').split()
    row = []
    for i in range(0, len(line), 3):
        pixel = line[i:i+3]
        row.append(pixel)
    image.append(row)
```

Get the width and height

How would you get the width and height here?



```
image = []
file_name = input('Enter file name: ')
ppm_file = open(file_name, 'r')
ppm_file.readline()
ppm_file.readline()
ppm_file.readline()

for line in ppm_file:
    line = line.strip('\n').split()
    row = []
    for i in range(0, len(line), 3):
        pixel = line[i:i+3]
        row.append(pixel)
    image.append(row)
```


Get the width and height

```
width = len(image[0])  
height = len(image)
```

```
image = []  
file_name = input('Enter file name: ')  
ppm_file = open(file_name, 'r')  
ppm_file.readline()  
ppm_file.readline()  
ppm_file.readline()
```

```
for line in ppm_file:  
    line = line.strip('\n').split()  
    row = []  
    for i in range(0, len(line), 3):  
        pixel = line[i:i+3]  
        row.append(pixel)  
    image.append(row)
```



What next?

After the width, height, and pixel info are available, what can be done?

What next?

After the width, height, and pixel info are available, what can be done?

- Display it!
- Process it!
- Implement a greenscreen!

Display the image

```
image = []
file_name = input('Enter file name: ')
ppm_file = open(file_name, 'r')
ppm_file.readline()
ppm_file.readline()
ppm_file.readline()

for line in ppm_file:
    line = line.strip('\n').split()
    row = []
    for i in range(0, len(line), 3):
        pixel = line[i:i+3]
        row.append(pixel)
    image.append(row)

width = len(image[0])
height = len(image)
```

```
image = []
```

```
. . .
```

```
width = len(image[0])
```

```
height = len(image)
```

```
gui = graphics(width, height, file_name)
```

```
for i in range(len(image)):
```

```
    for j in range(len(image[i])):
```

```
        # What goes here?
```

```
image = []
```

```
. . .
```

```
width = len(image[0])
```

```
height = len(image)
```

```
gui = graphics(width, height, file_name)
```

```
for i in range(len(image)):
```

```
    for j in range(len(image[i])):
```

```
        color = gui.get_color_string(int(image[i][j][0]),
```

```
                                     int(image[i][j][1]),
```

```
                                     int(image[i][j][2]))
```

```
        gui.rectangle(j, i, 1, 1, color)
```

What happens . . .

- When this program is run with a small image?

What happens . . .

- When this program is run with a small image?
- How could we change the program so that:
 - The user could specify a scale factor
 - The canvas size and displayed image would be zoomed in/out by the number the user specifies


```
width = len(image[0])  
height = len(image)
```

Add a scale factor

```
gui = graphics(width, height, file_name)
```

```
for i in range(len(image)):  
    for j in range(len(image[i])):  
        color = gui.get_color_string(int(image[i][j][0]),  
                                     int(image[i][j][1]),  
                                     int(image[i][j][2]))  
        gui.rectangle(j, i, 1, 1, color)
```

```
scale = float(input('Scale factor: '))
```

Add a scale factor

```
width = len(image[0]) * scale
```

```
height = len(image) * scale
```

```
gui = graphics(width, height, file_name)
```

```
for i in range(len(image)):
```

```
    for j in range(len(image[i])):
```

```
        color = gui.get_color_string(int(image[i][j][0]),
```

```
                                     int(image[i][j][1]),
```

```
                                     int(image[i][j][2]))
```

```
        gui.rectangle(j * scale, i * scale, scale, scale, color)
```