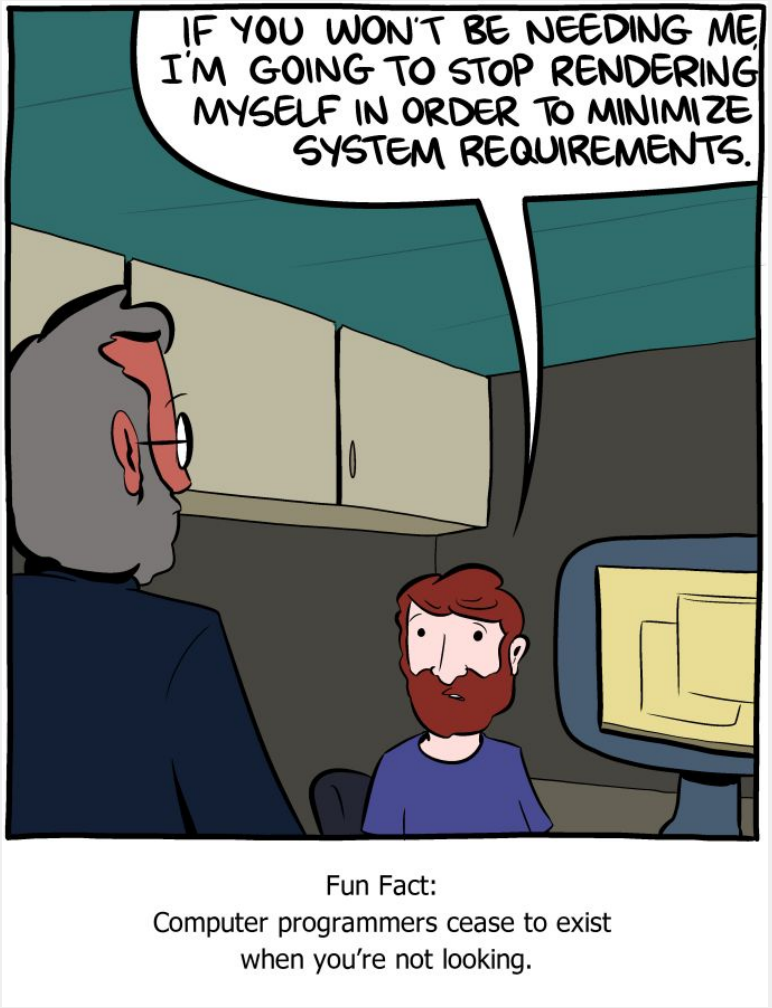
CS 110
Graphics and Motion

Benjamin Dicken

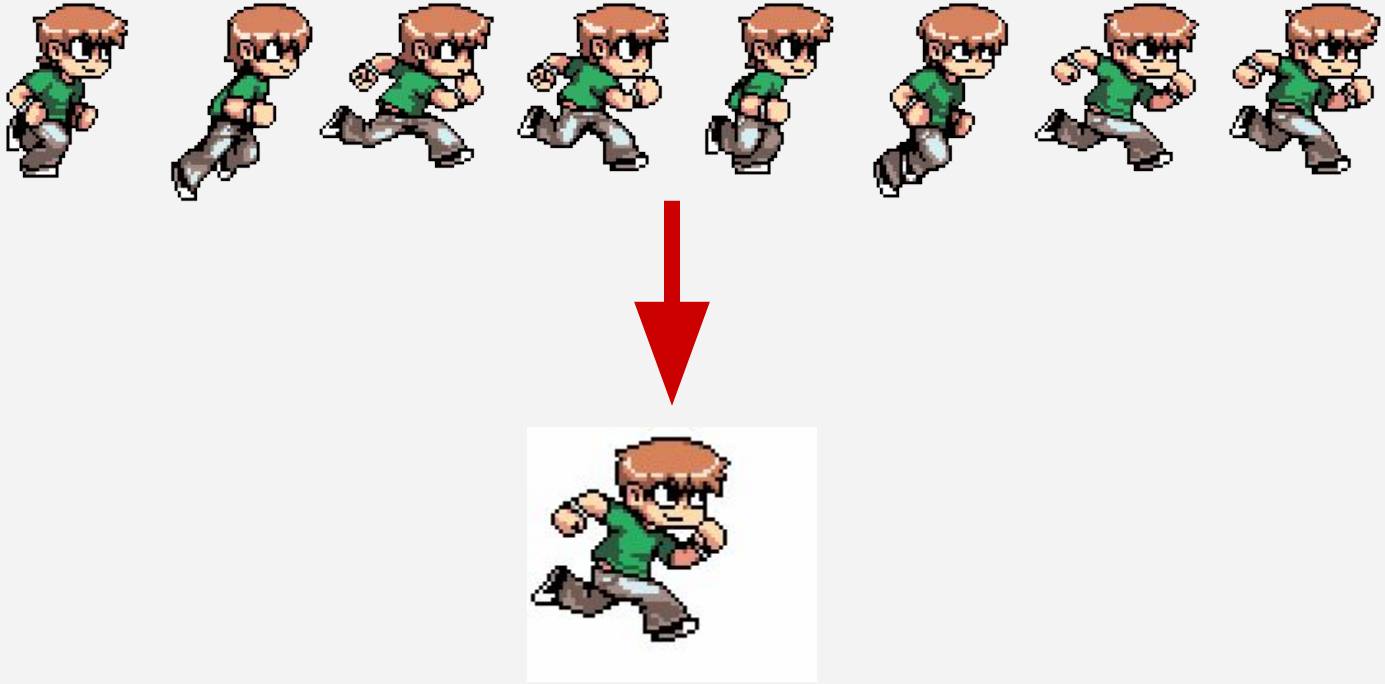


Motion

- When you watch a movie or play a video game and see motion, this is typically a bunch of images, displayed in quick succession
- This gives the perception of motion

- When writing a graphical program that has animation, we can do so with a **draw loop**

Motion

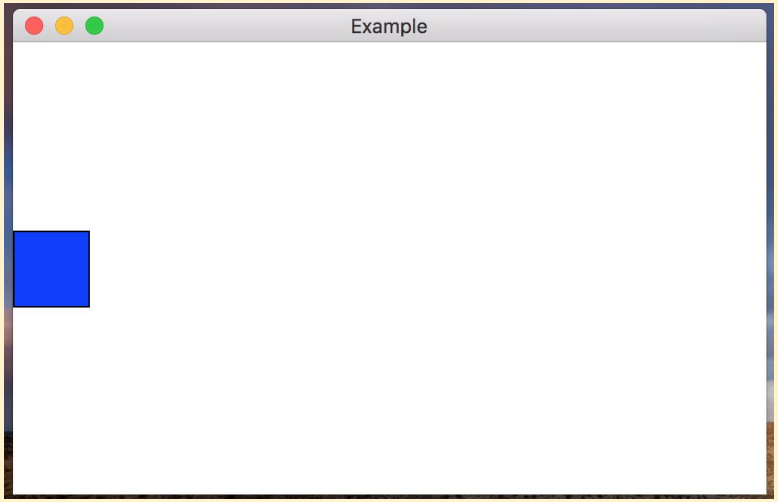


The general outline

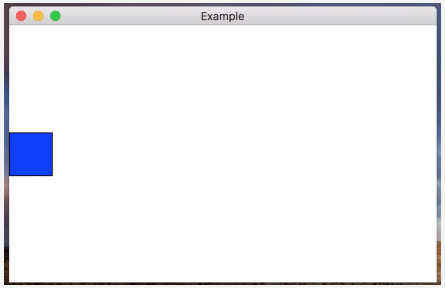
```
def main():
    gui = graphics(width, height, 'Sup?')
    while True:
        gui.clear()
        # Drawing code here
        gui.update_frame(30)

main()
```

Draw This, with a program that has a draw loop

```
def main():
    gui = graphics(width, height, 'Sup?')
    while True:
        gui.clear()
        # Drawing code here
        gui.update_frame(30)

main()
```



```
from graphics import graphics

def main():
    gui = graphics(500, 300, 'Example')
    while True:
        gui.clear()
        gui.rectangle(0, 125, 50, 50, 'blue')
        gui.update_frame(30)

main()
```



Make the rectangle move

Make the rectangle move

```
from graphics import graphics

def main():
    gui = graphics(500, 300, 'Example')
    while True:
        gui.clear()
        gui.rectangle(0, 125, 50, 50, 'blue')
        gui.update_frame(30)

main()
```

```
from graphics import graphics

def main():
    gui = graphics(500, 300, 'Example')
    x_coord = 0
    while True:
        gui.clear()
        gui.rectangle(x_coord, 125, 50, 50, 'blue')
        gui.update_frame(30)
        x_coord += 2

main()
```

Make the rectangle move, wrapping around

Make the rectangle move, wrapping around

```
def main():
    gui = graphics(500, 300, 'Example')
    x_coord = 0
    while True:
        gui.clear()
        gui.rectangle(x_coord, 125, 50, 50, 'blue')
        gui.update_frame(30)
        x_coord += 2


main()
```

```
from graphics import graphics

def main():
    gui = graphics(500, 300, 'Example')
    x_coord = -50
    while True:
        gui.clear()
        gui.rectangle(x_coord, 125, 50, 50, 'blue')
        gui.update_frame(60)
        x_coord += 10
        if x_coord > 550:
            x_coord = -50

main()
```

Make the rectangle move

Make the rectangle move

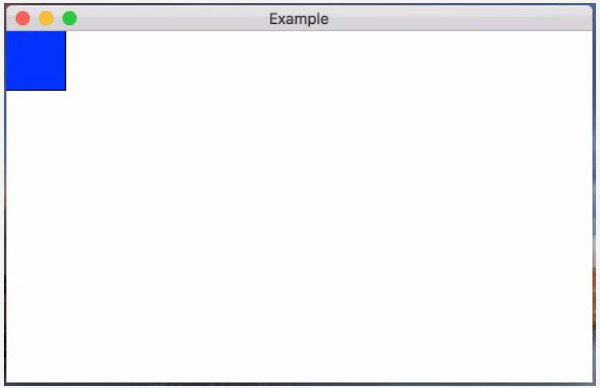
```
def main():
    gui = graphics(500, 300, 'Example')
    x_coord = 0
    while True:
        gui.clear()
        gui.rectangle(x_coord - 100, 125, 50, 50, 'blue')
        gui.update_frame(60)
        x_coord += 10
        if x_coord > 550:
            x_coord = -50
```

```
from graphics import graphics

def main():
    gui = graphics(500, 300, 'Example')
    x_coord = 0
    y_coord = 0
    while True:
        gui.clear()
        gui.rectangle(x_coord, y_coord, 50, 50, 'blue')
        gui.update_frame(30)
        x_coord += 5
        y_coord += 3

main()
```

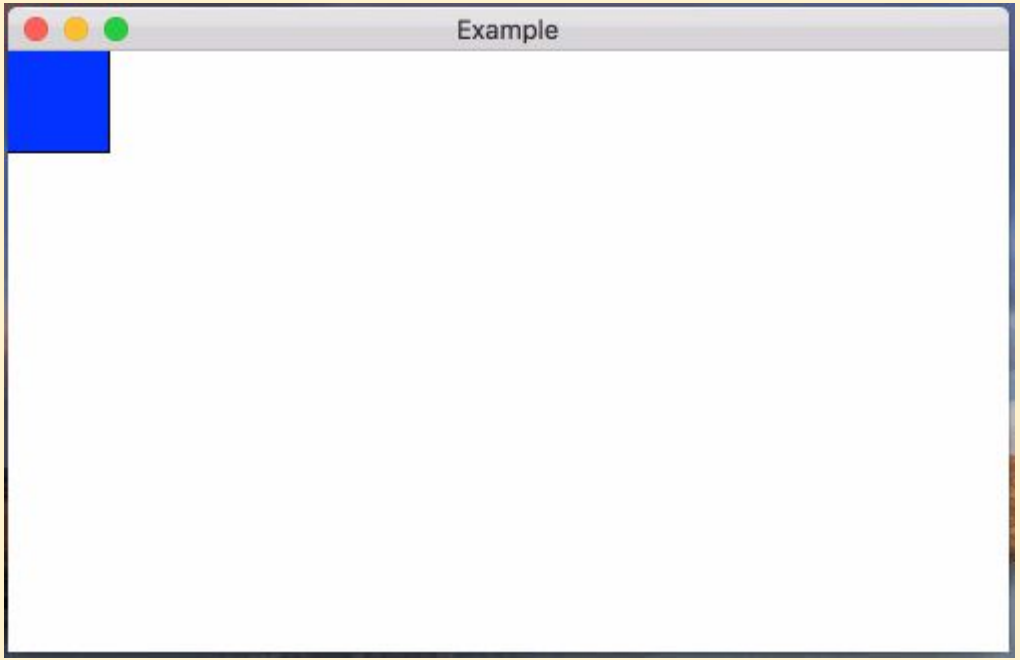
Mouse Position

- You can access the x and y coordinates of the mouse via the **graphics** object

```
graphics.mouse_x

graphics.mouse_y
```

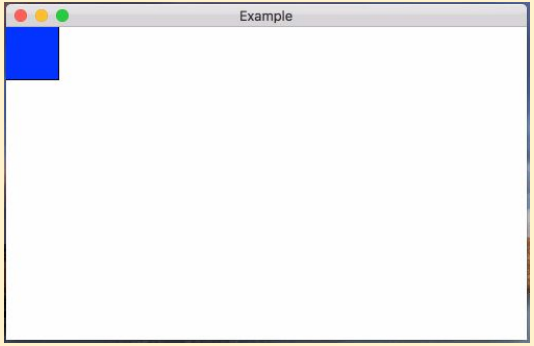


Make the square move with the mouse



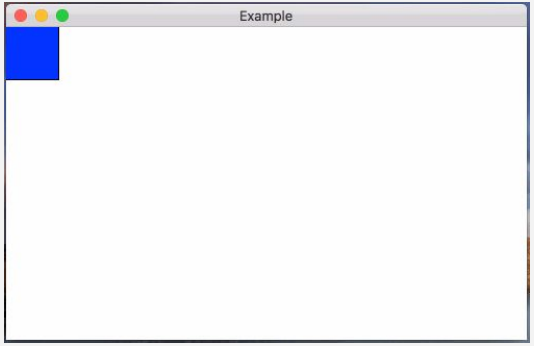
ICA

What should the x and y coordinates be?



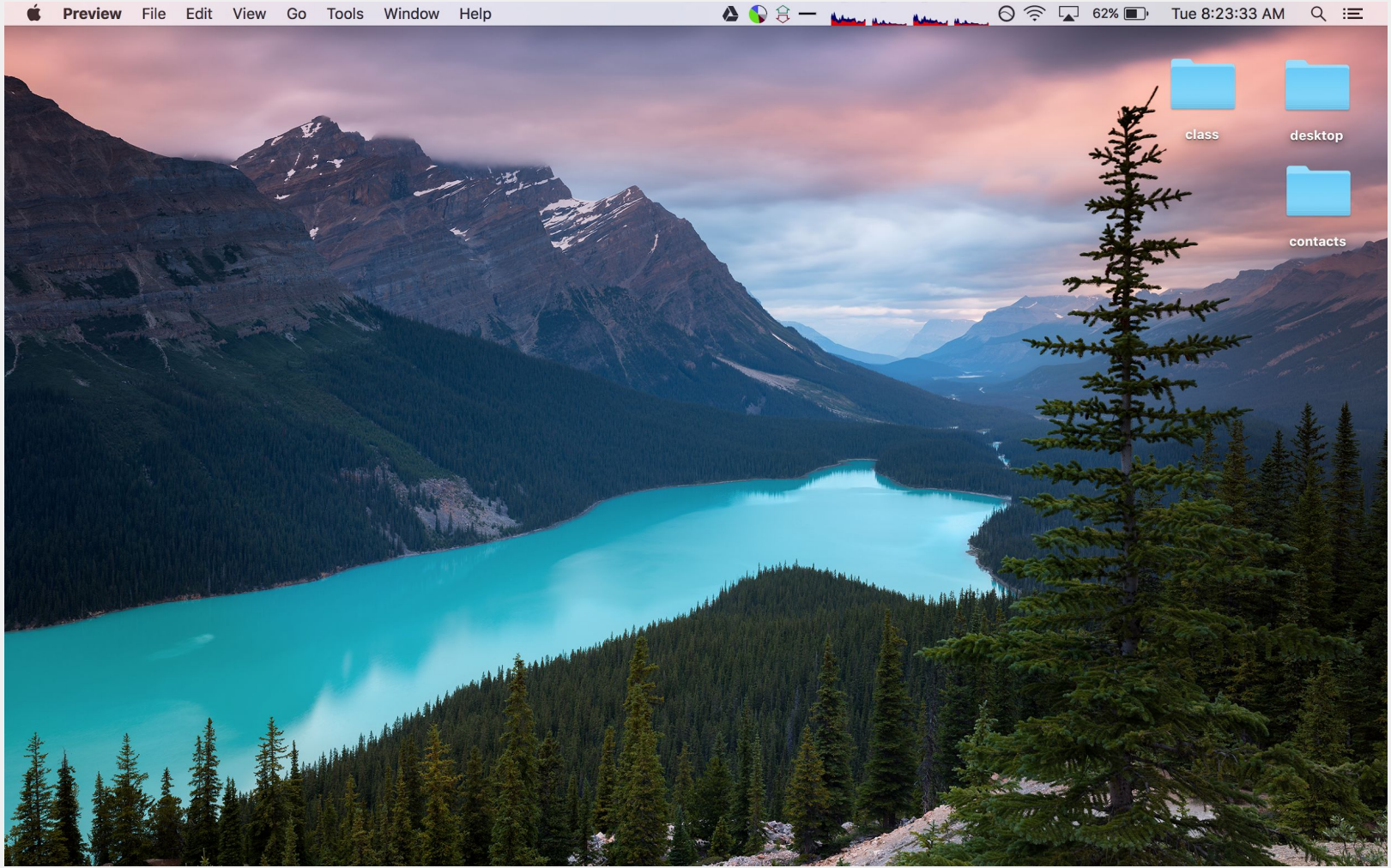

```
from graphics import graphics

def main():
    gui = graphics(500, 300, 'Example')
    while True:
        gui.clear()
        gui.rectangle(???, ???, 100, 100, 'blue')
        gui.update_frame(60)

main()
```

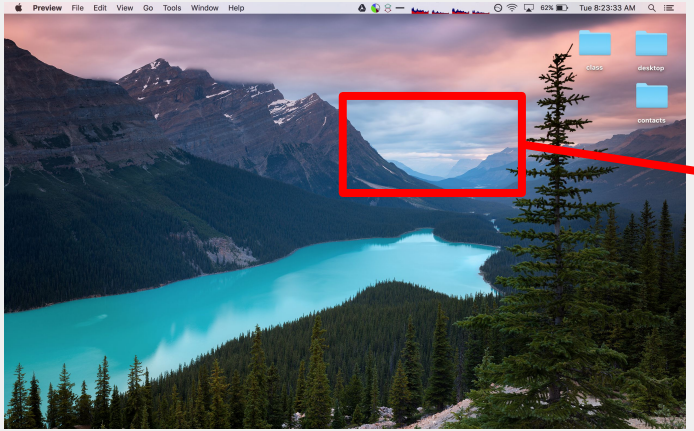
```
from graphics import graphics

def main():
    gui = graphics(500, 300, 'Example')
    while True:
        gui.clear()
        gui.rectangle(gui.mouse_x - 50, gui.mouse_y - 50, 100, 100, 'blue')
        gui.update_frame(60)

main()
```

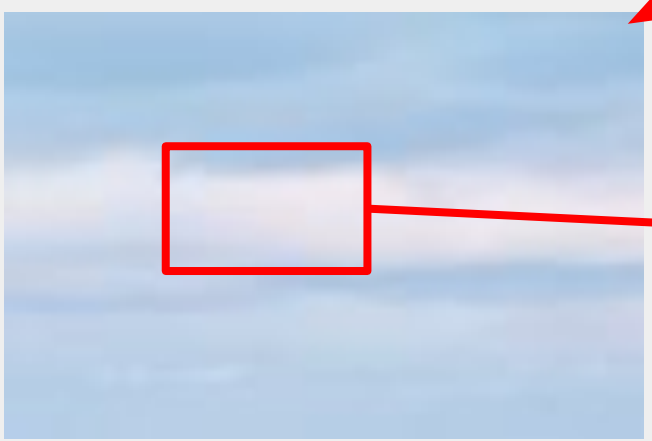


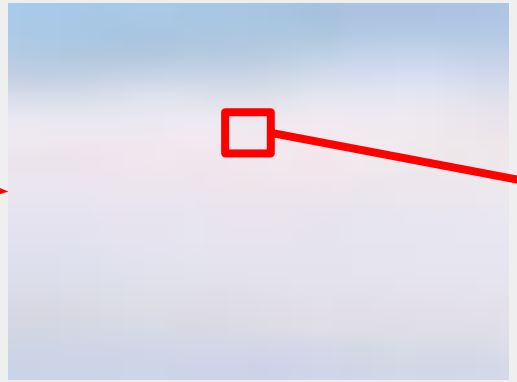
Color

- A computer screen is composed of a grid (rows and columns) of pixels
  - … Like the canvas of a processing window
- Each pixel can display a unique color, which is controlled by a **Red**, **Green**, and **Blue** brightness value


Preview
File
Edit
View
Go
Tools
Window
Help
62%
Tue 8:23:33 AM
class
desktop
contacts


Preview File Edit View Go Tools Window Help
Tue 8:23:33 AM
class
desktop
contacts





Color

- Thus, when we specify to a computer program, we can do so using these three numbers
- The brightness values for each can be between 0 (the darkest) and 255 (the highest)
    - 0-255 inclusive

Determine the colors

A. What color is (255, 0, 0) ?

B. What color is (100, 200, 255) ?

C. What color is (150, 200, 0) ?

Solutions

A. What color is (255, 0, 0) ?

B. What color is (100, 200, 255) ?

C. What color is (150, 200, 0) ?

Solutions

A. What color is (255, 0, 0) ?

B. What color is (100, 200, 255) ?

C. What color is (150, 200, 0) ?

Solutions

A. What color is (255, 0, 0) ?

B. What color is (100, 200, 255) ?

C. What color is (150, 200, 0) ?

What might the RGB values be?

A. Match this color: 

B. Match this color: 

C. Match this color: 

What might the RGB values be?

A. Match this color: 118, 253, 138

B. Match this color: 

C. Match this color: 

What might the RGB values be?

A. Match this color: 118, 253, 138

B. Match this color: 255, 251, 55

C. Match this color: 

What might the RGB values be?

A. Match this color: 118, 253, 138

B. Match this color: 255, 251, 55

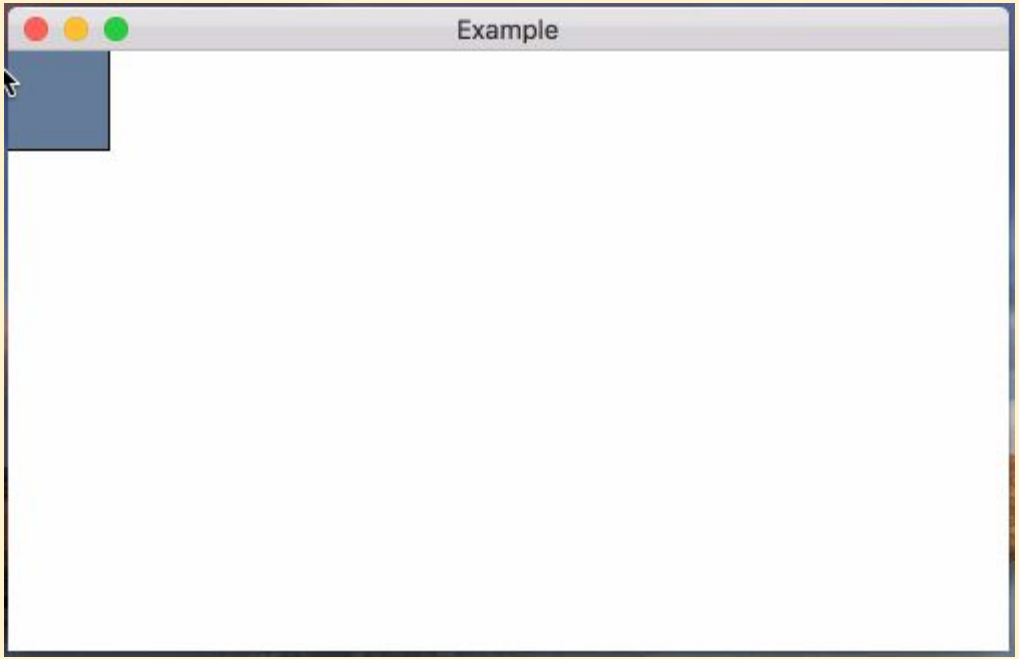
C. Match this color: 201, 253, 253

Getting a color string from RGB values

- The argument passed as the **fill** to the shape functions is expected to be a string
- How can we specify a color with RGB then?

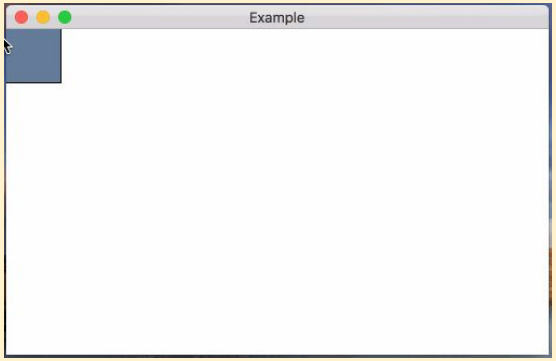
```
color_string = gui.get_color_string(100, 150, 200)
gui.rectangle(50, 50, 50, 50, color_string)
```

Make the color change roughly each second



ICA

Start from here!




```
from graphics import graphics

def main():
    gui = graphics(500, 300, 'Example')
    while True:
        gui.clear()
        gui.rectangle(gui.mouse_x - 50, gui.mouse_y - 50, 100, 100, 'blue')
        gui.update_frame(60)

main()
```

```
gui = graphics(500, 300, 'Example')
    color_string = 'blue'
    i = 0
    while True:
        if i % 60 == 0:
            red = random.randint(0, 255)
            green = random.randint(0, 255)
            blue = random.randint(0, 255)
            color_string = gui.get_color_string(red, green, blue)
        gui.clear()
        gui.rectangle(gui.mouse_x - 50, gui.mouse_y - 50, 100, 100, color_string)
        gui.update_frame(60)
        i += 1
```