Graphics

Chris Gregg
Based on slides by Chris Piech and Mehran Sahami
CS106A, Stanford University
Assignment 3: Images and Graphics!

• You can find assignment 3 on the CS 106A website
• It is a challenging assignment, but also a lot of fun
• There are two image problems (a warmup and a longer problem) and one graphics problem made up of a number of smaller problems.
• We will take some time now to go over the different problems.
• Don’t wait to start this assignment! Even though we have the diagnostic on Thursday, you should start the assignment now. Even though images and graphics won’t be on the diagnostic, the programming you will do for this assignment will give you more Python practice and will help you study.
Review
What are the basic operations needed to create a sepia tone? My reference point is the Perl Imagemagick library, so I can easily use any basic operation. I've tried to quantize (making it grayscale), colorize, and then enhance the image but it's still a bit blurry.

Simple code of a sepia converter in C# is available in my answer. What is wrong with this sepia tone conversion algorithm?

This algorithm comes from this page, each input pixel color is transformed in the following way:

```csharp
outputRed = (inputRed + .393) + (inputGreen + .769) + (inputBlue + .189)
outputGreen = (inputRed + .349) + (inputGreen + .686) + (inputBlue + .168)
outputBlue = (inputRed + .272) + (inputGreen + .534) + (inputBlue + .114)
```

If any of these output values is greater than 255, you simply set it to 255. These specific values are the values for sepia tone that are recommended by Microsoft.

You will need to use Math.Min likely. I tried doing the check for 255 after those three lines and an error will occur. I was facing the same problem earlier today when I was trying to make a sepia tone for my program. — BigBug Feb 28 '12 at 8:34

But what if I want something different to change the filter then how can I get to these values? Let my question is how we came to know about these values, do we need to just put different values again and again? — AHF Mar 23 '14 at 15:20
def main():
    image_name = input('enter an image name: ')
    image = SimpleImage('images/' + image_name)
    for pixel in image:
        sephia_pixel(pixel)
    image.show()

def sephia_pixel(pixel):
    R = pixel.red
    G = pixel.green
    B = pixel.blue
    pixel.red = 0.393 * R + 0.769 * G + 0.189 * B
    pixel.green = 0.349 * R + 0.686 * G + 0.168 * B
    pixel.blue = 0.272 * R + 0.534 * G + 0.131 * B
def main():
    image_name = input('enter an image name: ')
    image = SimpleImage('images/' + image_name)
    for y in range(image.height):
        for x in range(image.width):
            pixel = image.get_pixel(x, y)
            sepia_pixel(pixel)
    image.show()

def sepia_pixel(pixel):
    R = pixel.red
    G = pixel.green
    B = pixel.blue
    pixel.red = 0.393 * R + 0.769 * G + 0.189 * B
    pixel.green = 0.349 * R + 0.686 * G + 0.168 * B
    pixel.blue = 0.272 * R + 0.534 * G + 0.131 * B
def main():
    image_name = input('enter an image name: ')
    image = SimpleImage('images/' + image_name)
    for y in range(image.height):
        for x in range(image.width):
            pixel = image.get_pixel(x, y)
            sepia_pixel(pixel)
    image.show()

def sepia_pixel(pixel):
    R = pixel.red
    G = pixel.green
    B = pixel.blue
    pixel.red = 0.393 * R + 0.769 * G + 0.189 * B
    pixel.green = 0.349 * R + 0.686 * G + 0.168 * B
    pixel.blue = 0.272 * R + 0.534 * G + 0.131 * B
def main():

    for y in range(600):
        for x in range(800):
            print(x, y)
End Review
Today’s Goal

1. How do I draw shapes?
the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
    canvas.mainloop()
```
Draw a Rectangle

the following `main` method displays a blue square

def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
the following `main` method displays a blue square

def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
the following `main` method displays a blue square

def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
    canvas.mainloop()
```
the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```
the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```
the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```
the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100)
```

**Aside: Named Arguments**
This argument is named as `filled`. It allows functions to have arguments which you can ignore if you want a default value.
the following **main** method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")

Aside: Named Arguments
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.
Draw a Rectangle

the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
    canvas.mainloop()
```
Programming is Awesome!
- `canvas.create_line()`
- `canvas.create_oval()`
- `canvas.create_text()`
- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`
- `canvas.create_line(x1, y1, x2, y2)`

- `canvas.create_oval()`

- `canvas.create_text()`

The first point of the line is `(x1, y1)`
- `canvas.create_line(x1, y1, x2, y2)`

- `canvas.create_oval()`

- `canvas.create_text()`

The second point of the line is `(x2, y2)`
- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`
- canvas.create_line()
- canvas.create_oval(x1, y1, x2, y2)
- canvas.create_text()
- `canvas.create_line()`
- `canvas.create_oval(x1, y1, x2, y2)`
- `canvas.create_text()` (x1, y1)
- canvas.create_line()
- canvas.create_oval()
- canvas.create_text(x, y, text='hi')
• `canvas.create_line()`

• `canvas.create_oval()`

• `canvas.create_text(x, y, text='hi', anchor='w')`
Programming is Awesome

BY CHRIS PIECH

Graphics are really fantastic in python, especially using the TK library (which is the standard for Python). There are a lot of details, and as such a great way to learn is to look at worked examples.

Solution

```python
import tkinter
from PIL import ImageTk
from PIL import Image

CANVAS_WIDTH = 800
CANVAS_HEIGHT = 600

def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Awesome')
    # a line for good measure!
    canvas.create_line(0, 0, 600, 600)
    # a blue square with width and height = 80
    canvas.create_rectangle(70, 70, 150, 150, fill="blue")
    # a yellow rectangle that is long and skinny
    canvas.create_rectangle(620, 100, 640, 510, fill="yellow")
```

Handouts

- General Information
- Course Placement
- Honor Code
- Installing PyCharm
- Using Karel in PyCharm
- Submitting Assignments
- Diagnostic
- Image Reference

Graphics Reference
Goal

[Image: Checkerboard]
Milestone 1
Milestone 2
Milestone 3
Teaser for tomorrow...
def draw_square(canvas, row, col):

    If you get a copy when you pass a parameter. Does this copy the canvas??!!

Large variables are stored using something like a URL. The URL gets copied
How do you share google docs?

https://docs.google.com/document/d/1eBtEii13KHe
fFS-kSAOpXqeSXpbfTTMlmOgj6l9dvk/
```python
def main():
    canvas = make_canvas(...)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```
```python
def main():
    canvas = make_canvas(...)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```

Diagram:
- Stack:
  - main
  - canvas
- Heap:
  - memory.com/42
```python
def main():
    canvas = make_canvas(...)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```
def main():
    canvas = make_canvas(...)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```python
def main():
    canvas = make_canvas(...)  
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```

The diagram illustrates the stack and heap memory allocation:

- **Main function**
  - `main`
  - `canvas` with a reference to a `canvas` object

- **Draw square function**
  - `draw_square`
  - `canvas` with a reference to a `canvas` object

- **Memory heap**
  - `42` stored as a reference in the `canvas` object
```python
def main():
    canvas = make_canvas(...)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```

[Diagram showing stack and heap with variables and function calls]
```python
def main():
    canvas = make_canvas(...)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```

Diagram illustrating the stack and heap with a call graph.
def main():
    canvas = make_canvas(...)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
Large variable types are stored as memory addresses

(which are like memory URLs)