Week 10

CSC111 — Spring 2015

Dominique Thiébaut
dthiebaut@smith.edu
Next Few Lectures

• **Image Processing** with **Nested For-Loops**

• **Lists** can be Used to Solve Many Problems (Chap. 11)

• **Class Inheritance**
Image Geometry & Coordinate System

Scanning Images using Nested For-Loops

Sweep

How RGB Works

Python Code for Image Processing

Demo
Image Processing
• Different image types: jpg, png, **gif**, eps, svg, tiff, etc.

• Zelle `graphics.py` library compatible with **gif** images only.

• Jpg and png files can be **converted to gif** using Web services
  (e.g. [http://image.online-convert.com/convert-to-gif](http://image.online-convert.com/convert-to-gif))
Image Geometry

309 pixels $\times$ 163 pixels
$\sim$50K pixels
Image Geometry

309 pixels x 163 pixels
~50K pixels

width

height
Image Geometry
Image Geometry & Coordinate System

Scanning Images using Nested For-Loops

Sweep

How RGB Works

Python Code for Image Processing

Demo
Change all the pixels to red
WIDTH = 8
HEIGHT = 5

for x in range(WIDTH):
    for y in range(HEIGHT):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)

(Use numbers: easier to understand)
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed( x, y )
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

x=0
y=0

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)
x=0
y=0

WIDTH = 8
HEIGHT = 5

for x in range( 8 ):
    for y in range( 5 ):
        makePixelRed( x, y )
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)

x=0
y=1
WIDTH = 8
HEIGHT = 5

for \( x \) in range(8):
    for \( y \) in range(5):
        makePixelRed( \( x, y \) )

\( x=0 \)
\( y=2 \)
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)

x=0
y=2
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)

x=0
y=3
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)

x = 0
y = 3
WIDTH = 8
HEIGHT = 5

\[
\begin{align*}
  x &= 0 \\
  y &= 4
\end{align*}
\]

\[
\begin{align*}
  \text{for } x \ \text{in range}(8): \\
  &\quad \text{for } y \ \text{in range}(5): \\
  &\quad \quad \text{makePixelRed}(x, y)
\end{align*}
\]
WIDTH = 8
HEIGHT = 5

for $x$ in range(8):
    for $y$ in range(5):
        makePixelRed($x$, $y$)

$x$=0
$y$=4
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)

x=1
y=
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)

x = 1
y = 0
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

```python
for x in range(8):
    for y in range(5):
        makePixelRed(x, y)
```

$x=1$
$y=0X234$
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

for x in range(8):
    for y in range(5):
        makePixelRed(x, y)
Switching the Loops
WIDTH = 8
HEIGHT = 5

for y in range(5):
    for x in range(8):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

for y in range(5):
    for x in range(8):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

for y in range(5):
    for x in range(8):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

for y in range(5):
    for x in range(8):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

for y in range(5):
    for x in range(8):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

for y in range(5):
    for x in range(8):
        makePixelRed(x, y)
Image Geometry & Coordinate System

Scanning Images using Nested For-Loops

Sweep

How RGB Works

Python Code for Image Processing

Demo
WIDTH = 8
HEIGHT = 5

for y in range(5):
    for x in range(8):
        makePixelRed(x, y)
WIDTH = 8
HEIGHT = 5

for y in range( ? ):
  for x in range( ? ):
    makePixelRed( x, y )
WIDTH = 8
HEIGHT = 5

for y in range(4, -1, -1):
    for x in range(8):
        makePixelRed(x, y)

sweep!
Image Geometry & Coordinate System

Scanning Images using Nested For-Loops

Sweep

How RGB Works

Python Code for Image Processing

Demo
How RGB Works

pixel
How RGB Works

RGB System

Pixel

Red: 0-255
Green: 0-255
Blue: 0-255
How RGB Works

RGB System

Red | Green | Blue
-----|-------|-------
255  | 0     | 0     

pixel
How RGB Works

Pixel

RGB System

Red: 255  Green: 0  Blue: 0
How RGB Works

RGB System

Red Green Blue
0 255 0

pixel
How RGB Works

RGB System

pixel

<table>
<thead>
<tr>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>255</td>
<td>0</td>
</tr>
</tbody>
</table>
How RGB Works

RGB System

Pixel

Red  Green  Blue
0      0      255
How RGB Works

RGB System

<table>
<thead>
<tr>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>255</td>
</tr>
</tbody>
</table>
How RGB Works

RGB System

Red 51  Green 255  Blue 255
How RGB Works

RGB System

Red: 51
Green: 255
Blue: 255

http://www.rapidtables.com/web/color/RGB_Color.htm
How RGB Works

Red
51

Green
255

Blue
255

pixel at x, y

red, green, blue = cat.getPixel(x, y)
How RGB Works

Red | Green | Blue
---|---|---
51 | 255 | 255

# create color red
`color = color_rgb(255, 0, 0)`

# set pixel at x, y to red
`cat.setPixel(x, y, color)`
How RGB Works

<table>
<thead>
<tr>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

# create color red
color = color_rgb(255, 0, 0)

# set pixel at x, y to red
cat.setPixel(x, y, color)
Why 255?
binary

1 bit

0
1

decimal

0 - 1
binary

2 bits

00
01
10
11

decimal

0 - 3

\[ 2^2 = 4 \]
3 bits

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>111</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ 2^3 = 8 \]
4 bits

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

$2^4 = 16$

0 - 15
8 bits = byte

\[
\begin{align*}
00000000 & \quad 0 - 255 \\
00000001 & \\
00000010 & \\
00000011 & \\
00000100 & \\
00000101 & \\
00000110 & \\
\ldots & \\
11111000 & \\
11111001 & \\
11111010 & \\
11111011 & \\
11111100 & \\
11111101 & \\
11111110 & \\
11111111 & \\
\end{align*}
\]

\[2^8 = 256\]
8 bits = byte

00000000
00000001
00000010
00000011
00000100
00000101
00000110
...
11111000
11111001
11111010
11111011
11111100
11111101
11111110
11111111

0 - 255

Red  Green  Blue
51     255    255

1 pixel = 3 bytes
8 bits = byte

00000000
00000001
00000010
00000011
00000100
00000101
00000110
...
11111000
11111001
11111010
11111011
11111100
11111101
11111110
11111111

binary

decimal

0 - 255

Red
Green
Blue

1 pixel = 3 bytes

256 \times 256 \times 256 = 16,777,216 colors
Special Colors

- White:
  - Red: 255
  - Green: 255
  - Blue: 255

- Black:
  - Red: 0
  - Green: 0
  - Blue: 0

- Gray:
  - Red: 101
  - Green: 101
  - Blue: 101
Image Geometry & Coordinate System

Scanning Images using Nested For-Loops

Sweep

How RGB Works

Python Code for Image Processing

Demo
Displaying Images

The graphics module also provides minimal support for displaying and manipulating image PPM and GIF images. Display is done with an Image object. Images support the generic clone() method. Image-specific methods are given below.

**Image(anchrPoint, filename)**
Constructs an image from contents of the given file, centered at the given anchor point instead of filename. In this case, a blank (transparent) image is created of the given size.

**getAnchor()**
Returns a clone of the point where the image is centered.

**getHeight()**
Returns the height of the image.

**getWidth()**
Returns the width of the image.

**getPixel(x, y)**
Returns a list [red, green, blue] of the RGB values of the pixel at position (x, y); the intensity of the corresponding RGB color. These numbers can be turned into a color in a color module.

Note that pixel position is relative to the image itself, not the window where the image is always pixel (0, 0).

**setPixel(x, y, color)**
Sets the pixel at position (x, y) to the given color. Note: this is a slow operation.

**save(filename)**
Saves the image to a file. The type of the resulting file (e.g., GIF or PPM) is determined from the extension of filename. For example, img.save("myPic.ppm") saves img as a PPM file.

http://mcsp.wartburg.edu/zelle/python/graphics/graphics/node12.html
# A skeleton program to start doing image processing
# in Python
from graphics import *
# image geometry = 424x18
# make the window the same geometry
WIDTH = 424
HEIGHT = 418
IMAGEFILENAME = "catGlasses.gif"

def waitForClick( win, message ):
    """waitForClick: stops the GUI and displays a message.
    Returns when the user clicks the window. The message is erased."""

    # wait for user to click mouse to start
    startMsg = Text( Point( win.getWidth()/2, win.getHeight()-15 ), message )
    startMsg.draw( win )   # display message
    win.getMouse()         # wait
    startMsg.undraw()      # erase

def main():
    # open the window
    win = GraphWin( "Image Editor", WIDTH, HEIGHT )

    # open the cat image
    cat = Image( Point(WIDTH//2, HEIGHT//2), IMAGEFILENAME )
    cat.draw( win )

    waitForClick( win, "click to close" )

    win.close()

main()
def makeRed( win, img ):
    """ set the red component of all the pixels to 255, the maximum value. """
    global WIDTH, HEIGHT
    for x in range( WIDTH ):
        for y in range( HEIGHT ):
            red, green, blue = img.getPixel( x, y )
            img.setPixel( x, y, color_rgb(255, green, blue) )

def main():
    win = GraphWin( "Image Editor", WIDTH, HEIGHT )
    img = Image( Point(WIDTH//2, HEIGHT//2), IMAGEFILENAME )
    img.draw( win )
    makeRed( win, img )
    waitForClick( win, "click to close" )
    win.close()
Demo Time!
Transformations to Try

- Modify RED component
- Saturate
- Draw a horizontal line (beginning of a border)
We stopped here last time...
Mirroring an Image

Displaying a Checker Board

- 8x8 Grid
- Alternating Colors

Creating a Class for a Checkers Piece

Using a Gif Image for a Piece
Transformations to Try

- Mirror top half of cat image
Observation 1: $y_{\text{source}} + y_{\text{destination}} = 417$, always
Observation 2: \( 417 = \text{image height} - 1 \)
Mirroring an Image

Displaying a Checker Board

8x8 Grid

Alternating Colors

Creating a Class for a Checkers Piece

Using a Gif Image for a Piece
Graphic Problem of the Day: “Playing” Checkers

Image credit: http://www.freeimageslive.co.uk/free_stock_image/checkers.jpg
Problems to Solve:

- **Display** 8x8 board with alternating colors
- Generate the graphics for a **piece** (white and black circular shapes)
- **Display** the board with the black and white **pieces**
- Spiffy it up with **real images**
Display 8x8 Board
<table>
<thead>
<tr>
<th>i=0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>j=0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

side = 600/8 = 75 px

600px
\begin{itemize}
    \item \textit{i=0 1 2 3 4 5 6 7}
    \item \textit{j=0 1 2 3 4 5 6 7}
\end{itemize}

\begin{itemize}
    \item side = \frac{600}{8} = 75 \text{ px}
\end{itemize}

600px
# i is 0, j is 0

\[ p_1 = \text{Point}(?, ?) \]
\[ p_2 = \text{Point}(?, ?) \]

rect = \text{Rectangle}( p_1, p_2 )
rect.setFill( "green" )
rect.draw( win )

side = 600/8 = 75 px
# i is 1, j is 2

p1 = Point( ?, ? )
p2 = Point( ?, ? )

rect = Rectangle( p1, p2 )
rect.setFill( "green" )
rect.draw( win )
# i is 1, j is 2
x = i * side
y = j * side
p1 = Point( x, y )
p2 = Point( x+side, y+side )
rect = Rectangle( p1, p2 )
rect.setFill( "green" )
rect.draw( win )
for i in range(8):
    for j in range(8):
        x = i * side
        y = j * side
        p1 = Point(x, y)
        p2 = Point(x + side, y + side)

        rect = Rectangle(p1, p2)
        rect.setFill("green")
        rect.draw(win)
for i in range(8):
    for j in range(8):
        x = i * side
        y = j * side
        p1 = Point(x, y)
        p2 = Point(x+side, y+side)
        rect = Rectangle(p1, p2)
        rect.setFill("green")
        rect.draw(win)

For fun, replace with: color_rgb(i*j, 255-i*j, (i+j)*10)
Mirroring an Image

Displaying a Checker Board

8x8 Grid

Alternating Colors

Creating a Class for a Checkers Piece

Using a Gif Image for a Piece
### Alternating Black and White Cells

<table>
<thead>
<tr>
<th>0-0</th>
<th>1-0</th>
<th>2-0</th>
<th>3-0</th>
<th>4-0</th>
<th>5-0</th>
<th>6-0</th>
<th>7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>1-1</td>
<td>2-1</td>
<td>3-1</td>
<td>4-1</td>
<td>5-1</td>
<td>6-1</td>
<td>7-1</td>
</tr>
<tr>
<td>0-2</td>
<td>1-2</td>
<td>2-2</td>
<td>3-2</td>
<td>4-2</td>
<td>5-2</td>
<td>6-2</td>
<td>7-2</td>
</tr>
<tr>
<td>0-3</td>
<td>1-3</td>
<td>2-3</td>
<td>3-3</td>
<td>4-3</td>
<td>5-3</td>
<td>6-3</td>
<td>7-3</td>
</tr>
<tr>
<td>0-4</td>
<td>1-4</td>
<td>2-4</td>
<td>3-4</td>
<td>4-4</td>
<td>5-4</td>
<td>6-4</td>
<td>7-4</td>
</tr>
<tr>
<td>0-5</td>
<td>1-5</td>
<td>2-5</td>
<td>3-5</td>
<td>4-5</td>
<td>5-5</td>
<td>6-5</td>
<td>7-5</td>
</tr>
<tr>
<td>0-6</td>
<td>1-6</td>
<td>2-6</td>
<td>3-6</td>
<td>4-6</td>
<td>5-6</td>
<td>6-6</td>
<td>7-6</td>
</tr>
<tr>
<td>0-7</td>
<td>1-7</td>
<td>2-7</td>
<td>3-7</td>
<td>4-7</td>
<td>5-7</td>
<td>6-7</td>
<td>7-7</td>
</tr>
</tbody>
</table>

- **i - j**
### Alternating Black and White Cells

<table>
<thead>
<tr>
<th></th>
<th>0-0</th>
<th>1-0</th>
<th>2-0</th>
<th>3-0</th>
<th>4-0</th>
<th>5-0</th>
<th>6-0</th>
<th>7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>1-1</td>
<td>2-1</td>
<td>3-1</td>
<td>4-1</td>
<td>5-1</td>
<td>6-1</td>
<td>7-1</td>
<td></td>
</tr>
<tr>
<td>0-2</td>
<td>1-2</td>
<td>2-2</td>
<td>3-2</td>
<td>4-2</td>
<td>5-2</td>
<td>6-2</td>
<td>7-2</td>
<td></td>
</tr>
<tr>
<td>0-3</td>
<td>1-3</td>
<td>2-3</td>
<td>3-3</td>
<td>4-3</td>
<td>5-3</td>
<td>6-3</td>
<td>7-3</td>
<td></td>
</tr>
<tr>
<td>0-4</td>
<td>1-4</td>
<td>2-4</td>
<td>3-4</td>
<td>4-4</td>
<td>5-4</td>
<td>6-4</td>
<td>7-4</td>
<td></td>
</tr>
<tr>
<td>0-5</td>
<td>1-5</td>
<td>2-5</td>
<td>3-5</td>
<td>4-5</td>
<td>5-5</td>
<td>6-5</td>
<td>7-5</td>
<td></td>
</tr>
<tr>
<td>0-6</td>
<td>1-6</td>
<td>2-6</td>
<td>3-6</td>
<td>4-6</td>
<td>5-6</td>
<td>6-6</td>
<td>7-6</td>
<td></td>
</tr>
<tr>
<td>0-7</td>
<td>1-7</td>
<td>2-7</td>
<td>3-7</td>
<td>4-7</td>
<td>5-7</td>
<td>6-7</td>
<td>7-7</td>
<td></td>
</tr>
</tbody>
</table>

$i - j$
M mirroring an image
Displaying a Checker Board

8x8 Grid

Alternating Colors

Creating a Class for a Checkers Piece

Using a Gif Image for a Piece
Checkers on Board
Checkers on Board
Mirroring an Image

Displaying a Checker Board

8x8 Grid

Alternating Colors

Creating a Class for a Checkers Piece

Using a Gif Image for a Piece
Using an Image

http://pixgood.com/checkers-pieces-clip-art.html
Using an Image

http://pixgood.com/checkers-pieces-clip-art.html
# changeColorPiece.py
# D. Thiebaut
# Create new checkers image with different color.
# based on piece.gif. New image samed in piece2.gif
#
from graphics import *

WIDTH = 80  # dimension of checkers piece
HEIGHT = 80

def main():
    # open the window
    win = GraphWin( "Checkers", WIDTH, HEIGHT )

    # open image of black piece
    piece = Image( Point( WIDTH//2, HEIGHT//2 ), "piece.gif" )
    piece.draw( win )

    # change color of all the dark pixels
    for x in range( WIDTH ):
        for y in range( HEIGHT ):
            r, g, b = piece.getPixel( x, y )
            if r<10 and g<10 and b<10:
                continue
            newColor = color_rgb( r + (255-r)//2, g, b+(255-b)//2 )
            piece.setPixel( x, y, newColor )

    # save modified image into new file
    piece.save( "piece2.gif" )

    # close window
    win.close()

main()
We stopped here last time...
Review Wheel, Car, Checkers

Checkers: Animation Loop & Interactivity

A Virtual Aquarium
Review Wheel & Car Classes

http://cs.smith.edu/dftwiki/index.php/CSC111_Programs_for_Week_9_2015#wheel1.py
Review
DisplayCheckers.py

In “Programs for Week 10” page…
Review Wheel, Car, Checkers

Checkers: Animation Loop & Interactivity

A Virtual Aquarium
Building an Animation Loop
Adding Interactivity
Removing Checkers with the Mouse
Moving Checkers with the Mouse

D. Thiebaut, Computer Science, Smith College
Virtual Aquarium
# aquarium.gif
# D. Thiebaut
from graphics import *
import random

WIDTH = 700  # geometry of tank2.gif
HEIGHT = 517

def main():
    # open the window
    win = GraphWin( "CSC Aquarium", WIDTH, HEIGHT )

    # display background
    background = Image( Point( WIDTH//2, HEIGHT//2 ), "tank2.gif" )
    background.draw( win )

    win.getMouse()
    win.close()

main()