CSE1710

Week 11, Lecture 20

Fall 2013 ◆ Tuesday, Nov 19, 2013

Big Picture

This class meeting (L20) and the next one (L21) will be spent on Chapter 5 conceptsof the textbook.

There will be a labtest on Chapter 5 concepts on Thurs Nov 28/Fri Nov 29.

For the final three class meetings, and the final lab session we will be covering Chapter 6 concepts.

Image Recipes

"How to" guides: you want to iterate over…?

- …**all** of the pixels and mutate each one **unconditionally**
- …**all** of the pixels and mutate **some** of them **conditionally**
- …**some** of the pixels and mutate **some** of them **conditionally**

- There are a few skills here…
  - how to construct the loop you need
  - how to construct the boolean condition you need
- We will start with the basic case… but first a review of `Pixel` services

Pixels: How to mutate

We learned that a `Pixel` object encapsulates a few attributes:

- its parent picture
- its x and y location within its parent picture
- its colour

Of these three attributes…
- only one attribute is **mutable***: the pixel's colour.
- the pixel's (x,y) coordinate within its parent image **cannot** be changed; they are **immutable**

***mutable** means *able to be changed*

Pixels: How to mutate

Suppose the variable `p` is an object reference, and refers to a `Pixel` object.

So the only attribute I can change in a `Pixel` object is its colour…. here are some examples:

`p.setColor( … );`

this **must** be a reference to a `Color` object

construct one from scratch

use a predefined color

use a reference from somewhere else (e.g., another pixel)

Pixels: How to mutate

`p.setColor( … );`

this **must** be a reference to a `Color` object

construct one from scratch

use a predefined color

use a reference from somewhere else (e.g., another pixel)

`new Color(255, 0, 0)`

Color	Red	Green	Blue
🟥 Red	255	0	0
🟩 Green	0	255	0
🟦 Blue	0	0	255
🟨 Yellow	255	255	0
🟦 Cyan	0	255	255
🟪 Magenta	255	0	255
⬜ White	255	255	255
⬛ Black	0	0	0

Pixels: How to mutate

`p.setColor( … );`

this **must** be a reference to a `Color` object

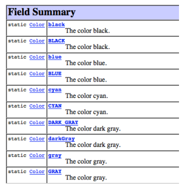
construct one from scratch

use a predefined color

use a reference from somewhere else (e.g., another pixel)

`Color.RED`

look at the API for the `Color` class…



Field Summary	
static Color	black The color black.
static Color	BLACK The color black.
static Color	blue The color blue.
static Color	BLUE The color blue.
static Color	cyan The color cyan.
static Color	CYAN The color cyan.
static Color	DARK_GRAY The color dark gray.
static Color	darkGray The color dark gray.
static Color	gray The color gray.
static Color	GRAY The color gray.

Pixels: How to mutate

`p.setColor( … );`

this **must** be a reference to a `Color` object

construct one from scratch

use a predefined color

use a reference from somewhere else (e.g., another pixel)

look at the API for the `Pixel` class to see the accessor method for an object's color attribute…

`r.getColor()`

Iterating over all of the pixels (v.1)

The first version of this involves treating the pixels as a collection and using the **collection-based version of iteration**

*not covered in Ch 5; if you like, read 8.2.4 for more background

```
Pixel[] thePixels = myPict.getPixels();
for (Pixel p : thePixels) {
      p.setColor(Color.RED);
}
```

Iterating over all of the pixels (v.2)

The second version of this involves iterating over a set of indices and using an **index-based accessor method to obtain a reference to each and every pixel**

There are two ways to do this: by index in the array of pixels and by row and column index.

First by index in the array of pixels

A Crash Course in Arrays

```
Pixel[] thePixels = myPict.getPixels();
int firstPixelIndex = 0;
int lastPixelIndex = thePixels.length-1;
```

`length` is a special **final** attribute of arrays in Java

WHY minus 1?

```
thePixels[60].getColor();
```

this retrieves the 60th element in the array

```
thePixels[lastPixelIndex+1].getColor();
```

What happens here?

```
thePixels[-1].getColor();
```

Iterating over all of the pixels (v.2)

the array of pixels…

so how could we automate this?

```
Pixel[] thePixels = myPict.getPixels();
// this sets the color of the first pixel
thePixels[firstPixelIndex].setColor(Color.RED);
// this sets the color of the second pixel
thePixels[firstPixelIndex+1].setColor(Color.RED);
//…
// this sets the color of the second-last pixel
thePixels[lastPixelIndex-1].setColor(Color.RED);
// this sets the color of the last pixel
thePixels[lastPixelIndex].setColor(Color.RED);
```

Iterating over all of the pixels (v.2)

```
Pixel[] thePixels = myPict.getPixels();
int firstPixelIndex = 0;
int lastPixelIndex = thePixels.length-1;

int currentIndex = firstPixelIndex;

for(; currentIndex <= lastPixelIndex; currentIndex++) {
       Pixel currentPixel = thePixels[currentIndex];
       currentPixel.setColor(Color.RED);
       myPict.repaint();
}
```

Comprehension Questions (1 of 6)

What happens in the code below:

```
Pixel[] thePixels = myPict.getPixels();
int firstPixelIndex = 0;
int lastPixelIndex = thePixels.length-1;

int currentIndex = firstPixelIndex;

for(; currentIndex <= lastPixelIndex; ) {
       Pixel currentPixel = thePixels[currentIndex];
       currentPixel.setColor(Color.RED);
       myPict.repaint();
       currentIndex++;
}
```

this is empty

Comprehension Questions (2 of 6)

What happens in the code below:

```
Pixel[] thePixels = myPict.getPixels();
int firstPixelIndex = 0;
int lastPixelIndex = thePixels.length-1;

int currentIndex = firstPixelIndex;

for(; currentIndex <= lastPixelIndex; ) {
       Pixel currentPixel = thePixels[currentIndex];
       currentPixel.setColor(Color.RED);
       myPict.repaint();
}
```

this is empty

Comprehension Questions (3 of 6)

What happens in the code below:

```
Pixel[] thePixels = myPict.getPixels();
int firstPixelIndex = 0;
int lastPixelIndex = thePixels.length-1;

int currentIndex = firstPixelIndex;

for(; true ; currentIndex++) {
       Pixel currentPixel = thePixels[currentIndex];
       currentPixel.setColor(Color.RED);
       myPict.repaint();
}
```

Comprehension Questions (4 of 6)

What happens in the code below:

```
Pixel[] thePixels = myPict.getPixels();
int firstPixelIndex = 0;
int lastPixelIndex = thePixels.length-1;

int currentIndex = firstPixelIndex;

for(; ; currentIndex++) {
        Pixel currentPixel = thePixels[currentIndex];
        currentPixel.setColor(Color.RED);
        myPict.repaint();
}
```

this is empty

17

Comprehension Questions (5 of 6)

What happens in the code below:

```
Pixel[] thePixels = myPict.getPixels();
int firstPixelIndex = 0;
int lastPixelIndex = thePixels.length-1;

int currentIndex = firstPixelIndex;

for(; currentIndex <= lastPixelIndex ;) {
        int x = currentIndex;
        currentIndex = -98923; //or any other crazy number
        Pixel currentPixel = thePixels[x];
        currentPixel.setColor(Color.RED);
        myPict.repaint();
        currentIndex = x+1;
}
```

this condition becomes **false** in the middle of the body

18

Comprehension Questions (6 of 6)

What happens in the code below:

```
Pixel[] thePixels = myPict.getPixels();
int firstPixelIndex = 0;
int lastPixelIndex = thePixels.length-1;

int currentIndex = lastPixelIndex+1;

for(; currentIndex <= lastPixelIndex ;) {
        Pixel currentPixel = thePixels[currentIndex];
        currentPixel.setColor(Color.RED);
        myPict.repaint();
}
```

19

The `DigitalPicture` class

We have been using the service `getPixels()`

```
Pixel[] thePixels = myPict.getPixels();
```

There is also `getPixel(int, int)`

```
Pixel aPixel = myPict.getPixel(6,7);
```

This will get the pixel located in column 6, row 7

suppose we iterate over the columns, and then for each column, we iterate over each row in that column...

20

Iterating over the columns

```
int firstColumn = 0;
int lastColumn = myPict.getWidth() - 1;

int currentIndex = firstColumn;

for (; currentIndex <= lastColumn; currentIndex++) {
    int numRows = myPict.getHeight();
    stdOut.printf("column #: %s has %s rows. %n", currentIndex, numRows);
    // the 1st pixel is myPict.getPixel(currentIndex, 0);
    // the 2nd pixel is myPict.getPixel(currentIndex, 1);
    // the 3rd pixel is myPict.getPixel(currentIndex, 2);
    // ...
    // the 2nd-last pixel is myPict.getPixel(currentIndex, numRows-2);
    // the last pixel is myPict.getPixel(currentIndex, numRows-1);
    // more generally…
    // Pixel thePixel = myPict.getPixel(currentIndex, i);
}
```

…and the rows

```
int firstColumn = 0;
int lastColumn = myPict.getWidth() - 1;

int currentIndex = firstColumn;

for (; currentIndex <= lastColumn; currentIndex++) {
    int numRows = myPict.getHeight();
    stdOut.printf("column #: %s has %s rows. %n", currentIndex, numRows);

    for (int i = 0; i < numRows; i++) {
        Pixel thePixel = myPict.getPixel(currentIndex, i);
    }
}
```

…and now do something…

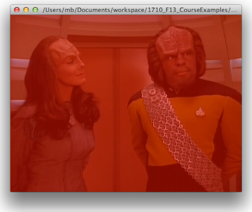
```
int firstColumn = 0;
int lastColumn = myPict.getWidth() - 1;

int currentIndex = firstColumn;

for (; currentIndex <= lastColumn; currentIndex++) {
    int numRows = myPict.getHeight();
    stdOut.printf("column #: %s has %s rows. %n", currentIndex,
    numRows);

    for (int i = 0; i < numRows; i++) {
        Pixel thePixel = myPict.getPixel(currentIndex, i);
        Color col = thePixel.getColor();
        stdOut.printf("colour of pixel (%s,%s) is %s. %n",
                                          currentIndex, i, col);
    }
}
```

mutate **every** pixel, column-by-column…

```
int firstColumn = 0;
int lastColumn = myPict.getWidth() - 1;

int currentIndex = firstColumn;

for (; currentIndex <= lastColumn; currentIndex++) {
    int numRows = myPict.getHeight();
    stdOut.printf("column #: %s has %s rows. %n", currentIndex,
    numRows);

    for (int i = 0; i < numRows; i=+1) {
        Pixel thePixel = myPict.getPixel(currentIndex, i);
        thePixel.setColor(Color.RED);
    }
}
```

this is equivalent to i++

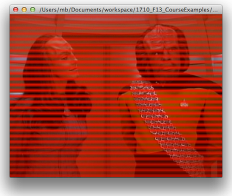
change every other column…



```
int firstColumn = 0;
int lastColumn = myPict.getWidth() - 1;

int currentIndex = firstColumn;

for (; currentIndex <= lastColumn; currentIndex+=2) {
    int numRows = myPict.getHeight();
    stdOut.printf("column #: %s has %s rows. %n", currentIndex,
    numRows);

    for (int i = 0; i < numRows; i=+1) {
        Pixel thePixel = myPict.getPixel(currentIndex, i);
        thePixel.setColor(Color.RED);
    }
}
```

YORK U UNIVERSITÉ UNIVERSITY

change every other row…



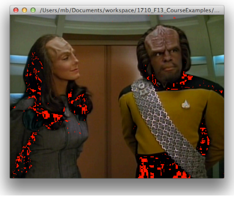
```
int firstColumn = 0;
int lastColumn = myPict.getWidth() - 1;

int currentIndex = firstColumn;

for (; currentIndex <= lastColumn; currentIndex+=1) {
    int numRows = myPict.getHeight();
    stdOut.printf("column #: %s has %s rows. %n", currentIndex,
    numRows);

    for (int i = 0; i < numRows; i=+2) {
        Pixel thePixel = myPict.getPixel(currentIndex, i);
        thePixel.setColor(Color.RED);
    }
}
```

YORK U UNIVERSITÉ UNIVERSITY

Test : is this pixel a shade of grey?

- if the RGB intensities are all the same
  - this gets perceived as shade of grey

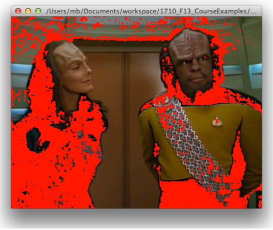
```
Pixel thePixel = myPict.getPixel(currentIndex, i);

boolean cond = thePixel.getRed() == thePixel.getGreen()
                              && thePixel.getRed() == thePixel.getBlue();
```



YORK U UNIVERSITÉ UNIVERSITY

Test : is this pixel **close** to a shade of grey?

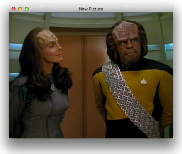
```
int THRES = 5;
Pixel thePixel = myPict.getPixel(currentIndex, i);
boolean cond = Math.abs(thePixel.getRed()-thePixel.getGreen()) < THRES
          && Math.abs(thePixel.getRed()-thePixel.getGreen()) < THRES;
```



YORK U UNIVERSITÉ UNIVERSITY

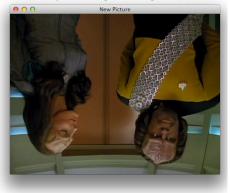
…make a copy, pixel by pixel

```
DigitalPicture myPict = new DigitalPicture(thePath);
DigitalPicture myPict2 = new DigitalPicture(myPict.getWidth(), myPict.getHeight());
myPict2.show();

int firstColumn = 0;
int lastColumn = myPict.getWidth() - 1;

int currentIndex = firstColumn;
int THRES = 5;

for (; currentIndex <= lastColumn; currentIndex += 1) {
        int numRows = myPict.getHeight();
        for (int i = 0; i < numRows; i += 1) {
                Pixel thePixel = myPict2.getPixel(currentIndex, i);
                Pixel theOrigPixel = myPict.getPixel(currentIndex, i);
                thePixel.setColor(theOrigPixel.getColor());
                myPict2.repaint();
        }
}

stdOut.println("Done.");
```



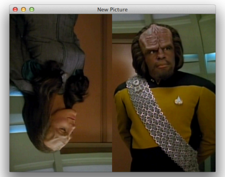
…make a flipped copy

```
DigitalPicture myPict = new DigitalPicture(thePath);
DigitalPicture myPict2 = new DigitalPicture(myPict.getWidth(),myPict.getHeight());
myPict2.show();

int firstColumn = 0;
int lastColumn = myPict.getWidth() - 1;

int currentIndex = firstColumn;
int THRES = 5;

for (; currentIndex <= lastColumn; currentIndex += 1) {
        int numRows = myPict.getHeight();
        for (int i = 0; i < numRows; i += 1) {
                Pixel thePixel = myPict2.getPixel(currentIndex, i);
                Pixel theOrigPixel = myPict.getPixel(currentIndex, numRows-1-i);
                thePixel.setColor(theOrigPixel.getColor());
                myPict2.repaint();
        }
}

stdOut.println("Done.");
```



…let's flip K'Ehleyr only

```
DigitalPicture myPict = new DigitalPicture(thePath);
DigitalPicture myPict2 = new DigitalPicture(myPict.getWidth(), myPict.getHeight());
myPict2.show();

int firstColumn = 0;
int lastColumn = myPict.getWidth() - 1;

int currentIndex = firstColumn;
int THRES = 5;

for (; currentIndex <= lastColumn; currentIndex += 1) {
        int numRows = myPict.getHeight();
        for (int i = 0; i < numRows; i += 1) {
                Pixel thePixel = myPict2.getPixel(currentIndex, i);
                int rowPos = i;
                if (currentIndex < lastColumn / 2) {
                        rowPos = numRows - 1 - i;
                }
                Pixel theOrigPixel = myPict.getPixel(currentIndex, rowPos);
                thePixel.setColor(theOrigPixel.getColor());
                myPict2.repaint();
        }
}
stdOut.println("Done.");
```


New Picture