CSE1710

Week 10, Lecture 19

Fall 2013 ◆ Thursday, Nov 14, 2013

YORK UNIVERSITÉ UNIVERSITY

Big Picture

For the next three class meetings, we will be covering Chapter 5 of the textbook. We will be using images to demonstrate the concepts of iterative and selection.

Reminder

On Thurs Nov 28/Fri Nov 29, we will have out **final** labtest.

YORK UNIVERSITÉ UNIVERSITY

Images

- Take good notes – there is relatively little material in the textbook; most of the material will be provided in lecture

3

YORK UNIVERSITÉ UNIVERSITY

To work with images, we need to:

1. work with the file system
2. work with the operating system's window manager and the platform's graphics hardware
3. understand colour models and image representation formats
4. understand the services of the `DigitalPicture` and the `Pixel` classes
5. iterate and construct conditions

REVIEW OF LECTURE12…

4

4

YORK UNIVERSITÉ UNIVERSITY

A Code Segment, deconstructed

```
String myPathName = File.separator + "Users" + File.separator + "mb"
        + File.separator + "images" + File.separator + "treefrog.jpg";

DigitalPicture pict1 = new DigitalPicture(myPathName);
```

VS

```
String myPathName = "/Users/mb/images/treefrog.jpg";

DigitalPicture pict1 = new DigitalPicture(myPathName);
```


File pathnames are system dependent

The file separator can be abstracted away as `File.separator`

- Windows Local File System (LFS):
  - `C:\USER\DOCS\LETTER.TXT`
- Windows Uniform Naming Convention (UNC)
  - `\\Server\Volume\File`
- Unix-like OS
  - `/home/user/docs/Letter.txt`

The `DigitalPicture` class

- provides services to create and to manipulate digital pictures

img

Class DigitalPicture

```
java.lang.Object
  └ img.DigitalPicture
```

All Implemented Interfaces:
AbstractDigitalPicture

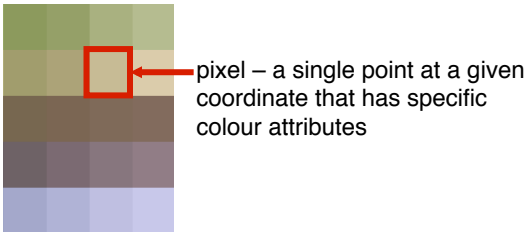
```
public class DigitalPicture
extends java.lang.Object
implements AbstractDigitalPicture
```

This class encapsulates a digital picture, which can be created from a file name, from another image, or from a width and height specification. A digital picture has accessor and mutator methods for its pixels. A digital picture has an associated string which serves as the picture's title. A digital picture can be displayed graphically. As well, some basic interaction with a digital picture is provided via the PictureExplorer. This class is based on the Picture class from Guzdial and Ericson, with explanatory comments and modifications provided by various contributors: sdc@cs.albany.edu, ericson@cc.gatech.edu. mb@cse.yorku.ca

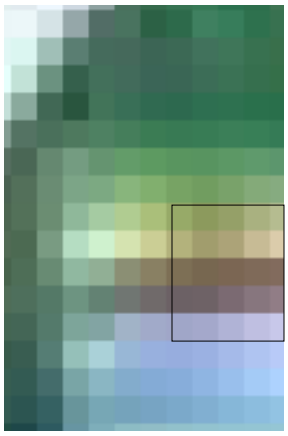
The `DigitalPicture` class a little more info

- attributes are:
  - `fileName : String` (might be `null`)
  - fileNameExtension : String (might be `null`)
  - `title : String`
  - `width : int`
  - `height : int`
  - `bufferedImage : BufferedImage`
    - the `BufferedImage` object encapsulates all of the pixels
  - the pixels are arranged in a **rectangular grid**

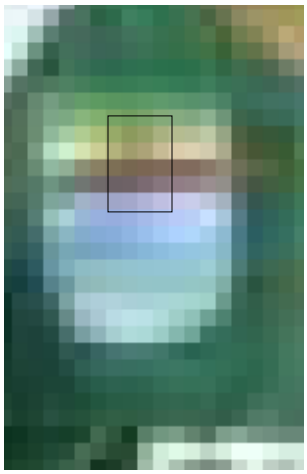
A rectangular grid of pixels



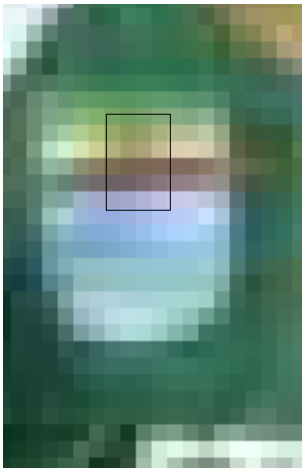
A rectangular grid of pixels



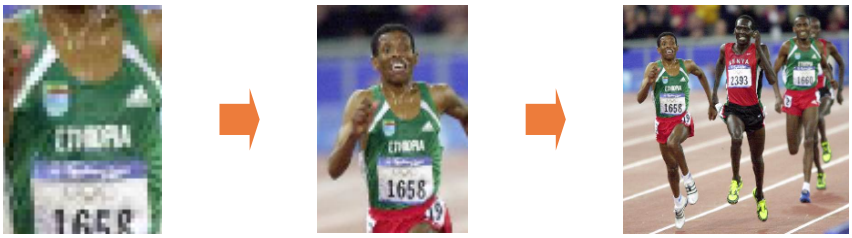
A rectangular grid of pixels



A rectangular grid of pixels



A rectangular grid of pixels



when the grid becomes large enough,
the human eye ceases to see the pixels as individual

A rectangular grid of pixels

The Rectangular Grid of Pixels

- each element has a `(x,y)` coordinate
  - the convention is that `(0,0)` is in the upper left hand corner
  - the *x* part of coordinate indicates the column
  - the *y* part of the coordinate indicates the row
  - ***in*** the door and ***down*** the stairs

The Rectangular Grid of Pixels

- the `DigitalPicture` class provides service to
  - get **all** of the pixels from an instance of a `DigitalPicture`
  - get a **specific** pixel from an instance of a `DigitalPicture`

The Rectangular Grid of Pixels

```
thePixel.setColor(new Color(255, 0. 0));
```

here we see the constructor for a instance of a object that encapsulates a particular colour that has RGB values of 255, 0, 0

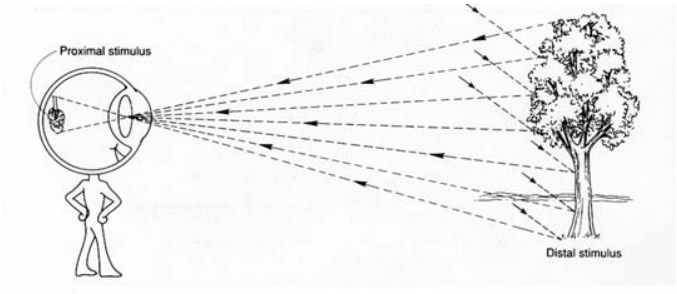
Small digression:

- Two Color Models: RGB and HSV
- The RGB model is much more intuitive than HSV
- We'll first explain RGB, then show the mapping into HSV space

- First, we will discuss the basics of vision…

The Retina

- the **retina** of the human eye is packed with photoreceptors
- the photoreceptors receive light stimulus via the lens of the eye

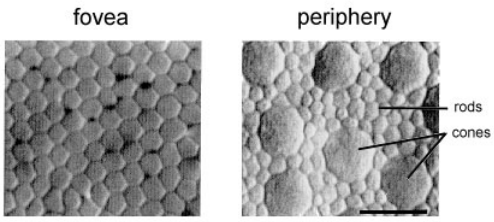


Areas of the Retina

- there are two types of **photoreceptors** in the **retina**
  - *rods*
  - *cones* … come in three types
    - short-wavelength
    - medium-wavelength
    - long-wavelength

the *fovea* is in the centre of the retina
rods: none
cones: completely and tightly packed

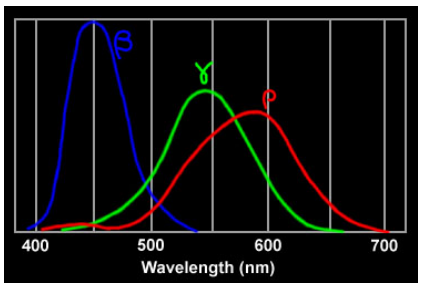
the *periphery* of retina
rods: more
cones: fewer
the proportion of rods to cones increase toward edge of retina



Hue

Hue corresponds to what we typically refer to as colour.
It is determined by the light's wavelength

Blue –
perceived by short-wavelength cones
Green –
perceived by medium-wavelength cones
Red –
perceived by long-wavelength cones



Specialized photoreceptors

- fovea
  - specialized for acute detailed vision
- periphery
  - does not provide acuity, but does detect change in scene (e.g., movement)
  - "something happened", but not what
  - rods are attuned to a broad spectrum of light
    - not specialized to particular wavelengths
    - more sensitive than cones (the threshold is lower)

Colour is complicated

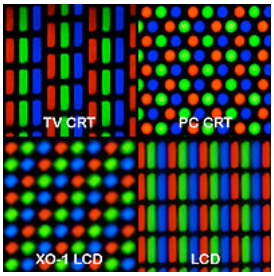
- perception based on 2 types of receptors (hue and intensity)
- our brain does more seeing than our eyes
- what we call colour is more accurately described as hue and brightness

A Key Fact

- the combination of red, blue and green is indistinguishable from **white** to the human eye
- this is exploited by computer displays

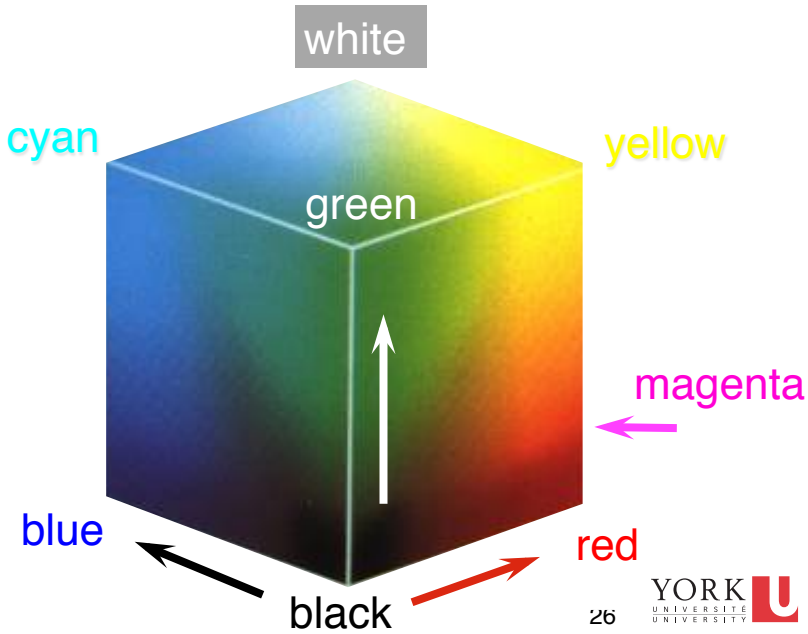
Pixels and Subpixels

- Many displays have a cluster of R, G, B sub-pixels for each pixel



- max *intensity* for R, G, B = seen as white
- min *intensity* for R, G, B = seen as black
- … and other saturated colours…

RGB Colour Space



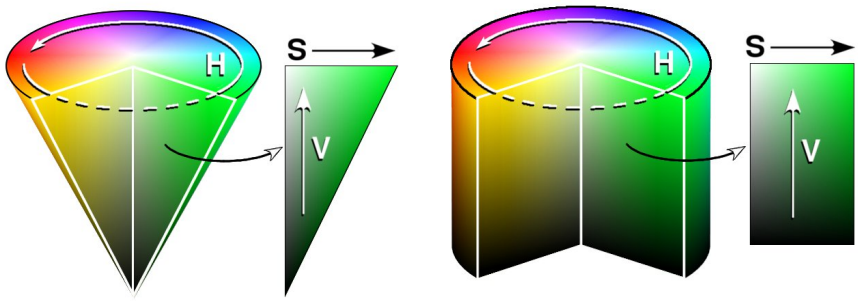
Color	Red	Green	Blue
Red	255	0	0
Green	0	255	0
Blue	0	0	255
Yellow	255	255	0
Cyan	0	255	255
Magenta	255	0	255
White	255	255	255
Black	0	0	0

Other cases…

- if the RGB intensities are all the same
  - this gets perceived as shade of grey
- if the RGB intensities are different
  - then perception depends on relative difference between strongest and weakest intensities
- Bottom line: Given a colour out in the world (that we see), it can be very difficult to determine the corresponding RGB values
  - typically easier to select via the HSV chooser

Hue-Saturation-Value (HSV) Model

- Each of hue, saturation, and brightness individually specified
- similarities to the way humans perceive and describe colour



Small digression:

- End of digression …. back to regular programming

Let's talk about two forms of iteration…

- one form: built upon a boolean condition
- another form: built around a *collection*

The "Collection" Form of Iteration

- a **collection** is simply a bunch of elements, possibly in a particular order, but not necessarily
- the **elements** must have a type (e.g., `int`, `Pixel`, etc)
- a **set** is a collection in which duplicates are not permitted
- a **list** is a collection in which the elements are ordered
- an **array** is a specific kind of list

collection, set, list	:	abstractions, not specific to Java
array	:	a Java programming element

The “Collection” Form of Iteration

```
for ( Type-of-Element  e : Identifier-of-Collection ) {
      // here is the body of the loop…
      }
}
```

FOR EXAMPLE:

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

// here we obtain an array

The “Collection” Form of Iteration

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

for (Pixel p : thePixels) {
     // here is the body of the loop…
}
```

The “Condition” Form of Iteration

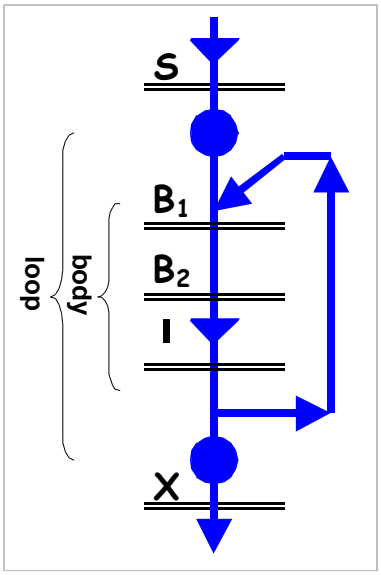
```
for (; boolean expression ;) {
     // here is the body of the loop…
}
```


The “Condition” Form of Iteration

```
for ( initial ; boolean expression ; bottom ) {
     // here is the body of the loop…
}
```

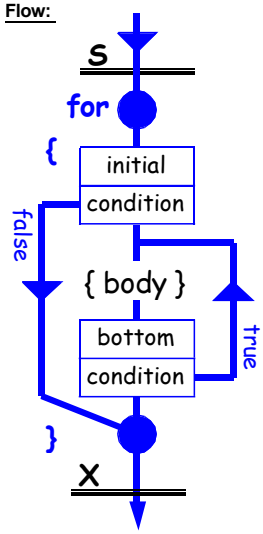
5.2.1 Flow of Control

Iteration



5.2.2 The for statement

Flow:



Syntax :

```
Statement -S
 for (initial; condition; b ottom)
 {
   body;
 }
Statement -X
```

Algorithm:

1. Start the `for` scope
2. Execute `initial`
3. If `condition` is `false` go to `9`
4. Start the body scope `{`
5. Execute the body
6. End the body scope `}`
7. Execute `bottom`
8. If `condition` is `true` go to `4`
9. End the `for` scope

Example

```
final int MAX = 10;
final double SQUARE_ROOT = 0.5;
for (int i = 0; i < MAX; i = i + 1)
{
   double sqrt = Math.pow(i, SQUARE_ROOT);
   output.print(i);
   output.print("\t"); // tab
   output.println(sqrt);
}
```


for (initial; condition; bottom)

```
for (int i = 0; i < MAX; i = i + 1)
{
   ...
}
```

```
int i;
for (; i < MAX; i = i + 1)
{
   ...
}
```

for (initial; condition; bottom)

- Can it be omitted?
- Can it be set to the literal true?
- What if it were false at the beginning?
- Is it monitored throughout the body?

for (initial; condition; bottom)

- Can it be any statement?
- Will the loop be infinite if it is omitted?