Media Computation Module

Last Updated: May 4, 2011

1 Module Name

Media Computation

2 Scope

Media Computation is a new type of introductory Computer Science class created to provide a path for those interested in doing creative, multimedia-related tasks with computing and attract them to the discipline of Computer Science. This module serves as a walkthrough tutorial to get students jump-started into what Media Computation is all about.

3 Learning Objectives

These are the learning objectives for this module:

a. Be able to read, understand, modify, and assemble from pieces of programs that achieve useful image manipulation tasks.

b. Gain familiarity with the programming language Python and the JES IDE.

c. Understand how images are represented in data and in what ways this data can be modified.

d. Grasp basic programming and computer science concepts like loops, parameters, variables, and functions.

4 5S characteristics of the module

a. Streams: this module deals with image manipulations. Images are represented as pixels, which are steams of data on a two dimensional space.

b. Structures: various algorithms, which have structures, are studied in this module.

c. Spaces: images are used as the main data in this module. They are represented on a two dimensional spaces. Therefore, this module has Spaces aspect.

d. Scenarios: software packages in this module have functions, which are to be applied in sequence to achieve tasks. These sequential steps are Scenarios.

5 Relationships with other modules

Module 3-b: Digitization can be taught before this Media Computation module considering that digitization is mentioned, but not explained enough in this module.

6 Perquisite knowledge/skills required

- General familiarity with Windows operating system in order to navigate Windows Server 2008 Datacenter:
  - Able to browse a file system.
  - Able to follow instructions to access an IBM Cloud using a Remote Desktop connection.

7 Introductory Remedial Instruction

None. This module itself serves at the remedial instruction so that a student knows what they are signing up for before getting deep into a Media Computation or introductory Computer Science course.

8 Body of Knowledge

Python is the programming language this class is based on. Specifically, the user will be working in Jython, which is a type of Python written with Java. Python is a high-level language and has extremely simple syntax as to emphasize readability, so it shouldn't overwhelm the users. Students will observe and write Python code through an extremely simple teaching IDE called JES (Jython Environment for Students). Students also will use these tools to manipulate images.

Python Basics:

- http://www.astro.ufl.edu/~warner/prog/python.html

Introduction to JES:

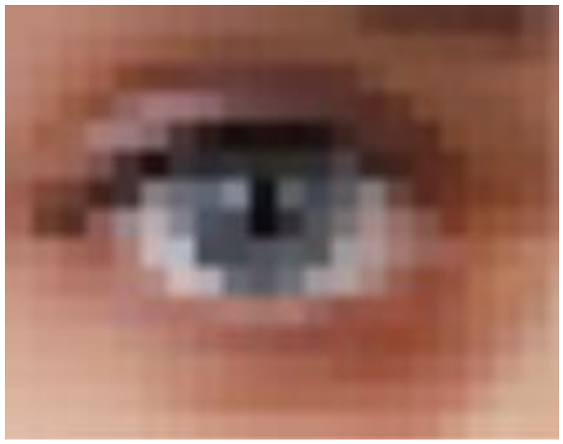
- http://alumni.cs.ucr.edu/~titus/JesIntro.pdf

Important topics include:

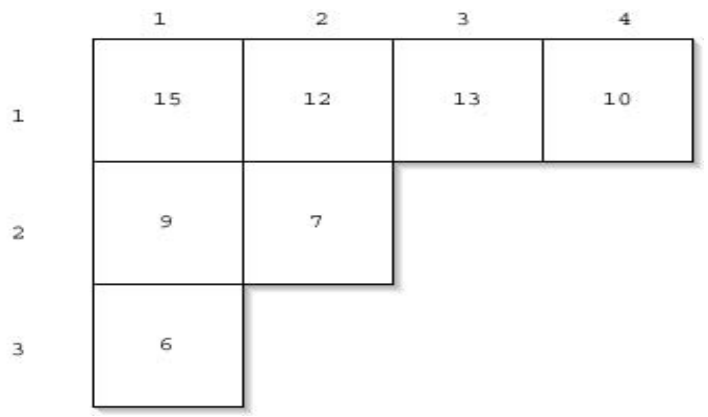
- Data representation
- Algorithms
- Encoding
- Pixels
- RGB values
- Loops
- Parameters
- Variables

Understanding images:

- We digitize pictures into lots of little dots
- Enough dots and it looks like a continuous whole to our eye
    - Our eye has limited resolution
    - Our background/depth acuity is particularly low
- Each picture element is referred to as a *pixel*



- How pixels are represented
    - Pixels are *picture elements*
    - Each pixel object knows its color
    - It also knows where it is in its picture
- A picture is a matrix of pixels
- It's not a continuous line of elements, that is, an *array*
- A picture has two dimensions: Width and Height
- We need a two-dimensional array: a *matrix*

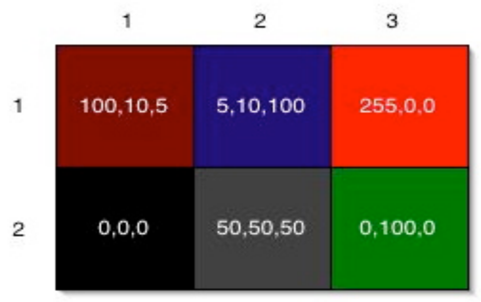


- We talk about positions in a matrix as (x, y), or (horizontal, vertical)
- Location (1,1) is the upper left corner

    - Element (2,1) in the matrix at left is the value 12
    - Element (1,3) is 6

- RGB
  - In RGB, each color has three component colors:
    - Amount of redness
    - Amount of greenness
    - Amount of blueness
  - Each does appear as a separate dot on most devices, but our eye blends them.
  - In most computer-based models of RGB, a single *byte (8 bits)* is used for each
  - So a complete RGB color is 24 bits, 8 bits of each

- Encoding RGB
  - Each component color (red, green, and blue) is encoded as a single byte
  - Colors go from (0,0,0) to (255,255,255)
  - If all three components are the same, the color is in gray scale
    - (50,50,50) at (2,2)
    - (0,0,0) (at position (1,2) in example) is black
    - (255,255,255) is white



	1	2	3
1	100,10,5	5,10,100	255,0,0
2	0,0,0	50,50,50	0,100,0

-JES image modifying package: Important functions
  - **getColor** takes a pixel as input and returns a Color object with the color at that pixel
  - **setColor** takes a pixel as input and a Color, then sets the pixel to that color
  - **makeColor** takes red, green, and blue values (in that order) between 0 and 255, and returns a Color object
  - **pickAColor** lets you use a color chooser and returns the chosen color
  - We also have functions that can **makeLighter** and **makeDarker** an input color
  - **getPixel** takes a coordinate and returns the pixel
  - **getPixels** takes a picture and returns its pixels as a matrix
  - **pickAFile** opens a browser from which the user can select a filename

9 Resources

Media Computation teacher's site:

- http://www.mediacomputation.org/

Media Computation @ Virginia Tech – Old student webpage:

- http://courses.cs.vt.edu/~cs1124/

Software Page:

- http://coweb.cc.gatech.edu/mediaComp-plan/94

10 Exercises/Learning activities

- Let's try writing some code
    - Log onto the cloud
    - Double click the JES application
- Type the command :
  >>>file=pickAFile()
- Hit enter and browse to the picture folder. Select image1.jpg.
- Now type:
  >>> pict=makePicture(file)
  >>> show(pict)
- You should see an image of Homer Simpson. We are now ready to make some changes to this picture.

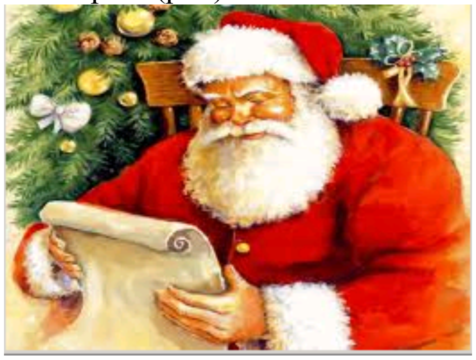
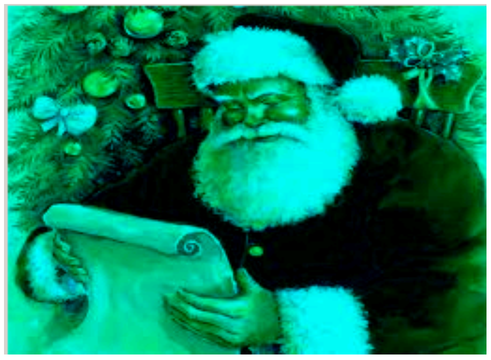
- Let's change a few pixels and see what happens:
  >>> setColor(getPixel(pict,10,11),black)
  >>> setColor(getPixel(pict,10,12),black)
  >>> setColor(getPixel(pict,10,13),black)
  >>> setColor(getPixel(pict,10,14),black)
- >>> repaint(pict)
- We can see the pixels we changed, but it isn't much
- To modify entire pictures in more interesting ways, we need to make use of a programming technique know as looping.
- Let's use a premade segment of code, or a program.
- Click File, Open program – go the Python Programs folder and select clearRed.
- clearRed should look like this:

```
def clearRed(picture):
   for pixel in getPixels(picture):
         setRed(pixel,0)
   return picture
```

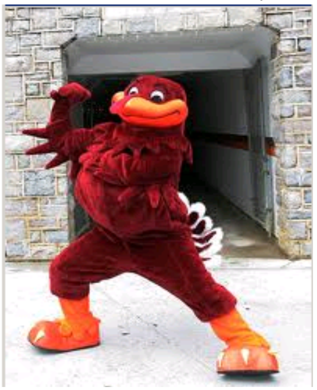
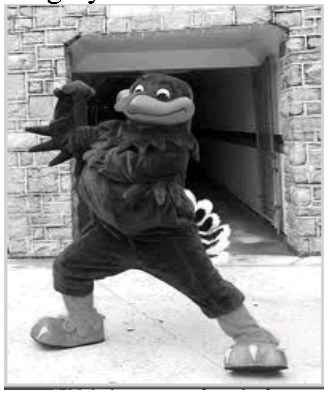
- The code in bold here is a *for loop*

FOR every pixel object **IN** the matrix of pixels for this picture, this program will set the pixels red value to zero.

- You can think of the clearRed program as a recipe that takes one ingredient, the picture to be modified.
- To use it we must first hit the Load Program button.
- Then we need to get a picture object for it to use.
  >>>file=pickAFile()

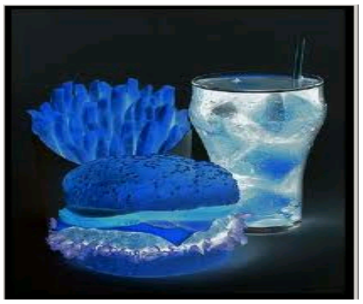
- Select image2.jpg this time.
```
>>> pict=makePicture(file)
>>> show(pict)
```
- We can use our loaded program like this.
```
>>> pict = clearRed(pict)
>>> repaint(pict)
```
 

- The beauty of programs like this clearRed function is that they can be used over and over again on all types of pictures.
- Just make sure you load the program and prepare a picture for the clearRed(picture) recipe and it will get rid of the red values.
- What if we wanted to make a picture look like a retro black and white photo? We would need to change each pixel to it's *gray scale* value.
- Load the program "greyscale" into JES and let's look at it.
- The red, green, and blue values of a pixel must be equal for the pixel to be grey.
- The average of all three values is a pixel's luminance, sort of like its brightness.
- If you set all values of a pixel to its luminance, you get its equivalent greyed-out version.
```
redness=getRed(p)
greenness=getGreen(p)
blueness=getBlue(p)
luminance=(redness+blueness+greenness)/3
setColor(p, makeColor(luminance,luminance,luminance))
```
- As you can see, only the brightly colored HokieBird changed significantly. The grey hokiestone behind him was already close to its greyscale value.

- Now let's try open another interesting program called "negative."
```
def negative(picture):
    for px in getPixels(picture):
            red=getRed(px)
            green=getGreen(px)
            blue=getBlue(px)
            negColor=makeColor(255-red,255-green,255-blue)
            setColor(px,negColor)
    return picture
```

- Can you guess what effect this will have?
- Let's say Red is 10. That's very light red.
    - What's the opposite? LOTS of Red!
    - The negative of that would be 245: 255-10
    - So, for each pixel, if we negate each color component in creating a new color, we negate the whole picture.



- Now let's open the last program, "blendTwoPictures"
- Take a careful look and see if you can figure out how it works. (Hint: There is one for loop inside of another, also called a *nested loop*)
- To run this program, note that the recipe calls for two file names instead of picture object. This is actually easier than the other recipes:

```
>>> fileName1 = pickAFile();
>>> fileName2 = pickAFile();
>>> show(blendTwoPictures(fileName1, fileName2))
```

- Try any two images you want! The results are always pretty cool.

- Can you guess what would happen if you blended a picture with its negative? (You should have a blank canvas, as they cancel each other out)

You have finished you introduction to the world of Media Computation and Computer Science!

11 Evaluation of learning objective achievement

After completed this module, the users should be able to write their own basic Python functions to modify images in simple, but powerful ways. The students should recognize media-manipulating Python code and be able to decipher what it is trying to accomplish. Users should also be comfortable with the JES IDE and be able to describe how images are represented and changed through computing.

12 Glossary

None

13 Additional useful links

Python Basics:
- http://www.astro.ufl.edu/~warner/prog/python.html

Introduction to JES:
- http://alumni.cs.ucr.edu/~titus/JesIntro.pdf

Official Python Website
- http://www.python.org

14 Contributors

Prepared for class CS4624 at Virginia Tech

- Initial Authors: Team 6
  - Dylan Slack

- Reviewer
  - Professor Steve Harrison
  - Seungwon Yang