Media Computing: Manipulating Pixels

A Picture is worth a Thousand Words Pixels!

An image is a matrix of pixels.

Below is a snapshot of an image being viewed JES.

The left top corner of the image is coordinate x=1, y=1. The bottom right corner is x=635, y = 740.

A pixel is a color. Colors are defined with three attributes: how much red, how much blue, and how much green.

\[ \text{Pixel} \rightarrow \{\text{redValue}, \text{GreenValue}, \text{BlueValue}\} \]

A pixel is thus an example of a complex data type, an object. In this case, the object has three parts.
Each of the RGB attributes are between 0 and 255 (how much data is needed to store such an attribute?)

{0,0,0} is black.

{255,255,255} is white

What is Red? Blue? Green?

Here's how you can explore an image, including the value of each pixel, in the JES environment:

1. Load an image using the MakePicture function:

   filename=pickAFile()
   pic=MakePicture(filename))

2. Choose Media Tools | Picture Tool. When the picture appears, click on it. The environment will display the x–y coordinate and RGB values for each pixel. This can help you find the RGB value of colors you want to use.

**Manipulating Pixels**

Here are some key functions provided by the JES environment:

- **makeColor(r,g,b)** → this returns a color, e.g.,
  
  white=makeColor(255,255,255)

- **getPixel(pic,x,y)** → this returns a pixel from an image, e.g.,
  
  pixel=getPixel(myPic,100,100)

Generally, you'll call setColor after calling getPixel:

- **setColor(pixel,color)** --> modify given pixel

The following changes the pixel at 100,100 to white:

- white=makeColor(255,255,255)
- pixel=getPixel(myPic,100,100)
- setColor(pixel,white)
You can also change the R, G, or B value of a color using the set/get color functions:

```plaintext
getRed(pixel)          setRed(pixel, value)
getGreen(pixel)        setGreen(pixel, value)
getBlue(pixel)         setBlue(pixel, value)
```

Use these when you want to see what color a pixel is and then change it, sort of like an $x=x+1$, e.g., the following will redden a pixel:

```plaintext
red = getRed(pixel)
setRed(pixel, red+20)  # be careful here for overflow
```

After making changes to a pic, you need to redraw it to see it:

```plaintext
repaint(pic) --> gets the system to redraw the picture.
```

With these functions, and what you know about loops, you can code some powerful image manipulations.

Here's a program that draws a white rectangle in the top-left corner:

```plaintext
name = pickAFile()
pic = makePicture(name)
show(pic)
color = makeColor(255,255,255)  # white
max = 100
x = 1
while x < max:
    y = 1
    while y < max:
        pixel = getPixel(pic, x, y)
        setColor(pixel, color)
        y = y + 1
    x = x + 1
repaint(pic)
```

Note that the above program draws in a fixed part of an image (100x100).
You can also perform operations based on the actual size of a JES picture by using:

- `getWidth(pic)`  \(\rightarrow\) this returns the width of the image.
- `getHeight(pic)`  \(\rightarrow\) this returns the height of the image

Using these methods allows you to process an entire image pixel-by-pixel. The loop template for this is:

\[
\begin{align*}
x &= 1 \\
\text{while } x < \text{getWidth(pic)}: \\
\quad y &= 1 \\
\quad \text{while } y < \text{getHeight(pic)}: \\
\quad\quad \text{pixel} &= \text{getPixel(pic},x,y) \\
\quad\quad \# \text{ do something to pixel} \\
\quad\quad y &= y + 1 \\
\quad x &= x + 1 \\
\end{align*}
\]

Be very careful with your indentation!

You’ll use loops like the above many times as you write the solutions to project 1.

**In-Class Assignment**

1. Type in the ‘white rectangle’ code on the previous page and get it to run.
2. Write a JES program ‘redEvenRows.py’ that draws the even rows of a picture red. Start with the template above. Test your program on at least two pictures.
3. Modify the program so that it draws every fourth column green.