1 Administrative Topics

- Don’t forget Bruce’s 1/2 hr rule. This applies if you have been working with a TA for 1/2 hr. Email him! (or me, but he is more likely to be awake late at night)

- Start your project early! If you need help, remember I have office hours on Fri, Mon, and Tues. I was very happy with how many people took advantage of my office hours yesterday afternoon.

2 How do objects fit into our memory model?

Recall the last function from Monday,

```python
# Change each pixel’s color to its "negative" value
# input: pm (<Pixmap>)
# output: None

def negatePixmap(pm):
    w = pm.getWidth()
    h = pm.getHeight()
    for x in range(w):
        for y in range(h):
            (r, g, b) = pm.getPixel(x, y)
            pm.setPixel(x, y, (255 - r, 255 - g, 255 - b))
```
and main code that calls it:

```python
if __name__ == '__main__':
    pm = graphics.Pixmap('test.ppm')
    negatePixmap(pm)
    displayPixmap(pm)
```

Let's put this code into a function named main1, changing the end of the file to look like this:

```python
def main1():
    pm = graphics.Pixmap('test.ppm')
    negatePixmap(pm)
    displayPixmap(pm)

if __name__ == '__main__':
    main1()
```

Let's look at the symbol tables for main1 and negatePixmap.

Let’s step through main1. The symbol table starts empty (there are no parameters). The first statement is an assignment statement (line 2). We evaluate the rhs and a Pixmap is returned. We then update the symbol table by adding an entry whose name is pm and whose value is an arrow (in the table) to a Pixmap object (outside the table):

![Symbol Table for main1](image)

Next, we call negatePixmap (line 3).

A new symbol table is created. There is an entry automatically added for the parameter. The name is pm and the value is an arrow to the same Pixmap object:
Then, after we execute lines 5 and 6, w and h are added:
Then, Python sets everything up for the nested for loops. First, the range is evaluated for \( x \), and \( x \) is set to the first value in that range. Then, the first line that must be executed in the \( x \)-loop is the \( y \)-loop, so Python evaluates the range for the \( y \) loop and sets \( y \) to the first value in its range. This means \( x \) and \( y \) are added to the symbol table, along with two hidden variables containing the ranges:

The first time through the inner loop (the \( y \) loop), \( r \), \( g \), and \( b \) are added to the symbol table. They contain the red, green, and blue values for the pixel at position \((0,0)\) (the upper left hand corner of the Pixmap). Then, the color of that pixel is changed (indicated by the color change in the Pixmap type):
Then, y is incremented, r, g, and b are updated, and the second pixel in the first column has its color changed:
And the inner block of code is repeated a total of 550*450 times. The pixels are edited column by column (because y is in the inner loop). At the end of the negatePixmap functions, the symbol tables loop like this:
Since both the main function and negatePixmap are pointing to the same object, there is no need for negatePixmap to return anything. It is simply manipulating the same pixels main1 has access to. This is helpful because it cuts down on the amount of memory required and on the time needed to copy data from one memory location to another. So, negativePixmap just finishes and its symbol table goes away. Control returns to main1 and it is now pointing to the updated Pixmap:
It is this new Pixmap that is passed in to displayPixmap.

Done with this demo.

What if we don’t want to alter the original pixmap? What if we want to keep it around so that we can apply a different manipulation function? One option is to create two Pixmap objects from the same file. Another is to use the clone method. pm.clone() will return a new copy of the Pixmap contained in pm.

Let’s write main2, based on main1, but which clones the Pixmap.

```python
1 def main2 ():
2     orig_pm = graphics.Pixmap(‘test.ppm’)
3     pm = orig_pm.clone()
4     negatePixmap(pm)
5     displayPixmap(pm)
```

After executing line 2, the symbol table for main2 has a single entry containing the Pixmap:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>orig_pm</td>
<td></td>
</tr>
</tbody>
</table>
After executing line 3, the symbol table references the original Pixmap and a clone:

After executing line 4, the symbol table references one unedited Pixmap, and one negative Pixmap: