

CMSC 143: Introduction to Object-Oriented Programming with Robots

Lab 10: Image Expressions

Due May 2, 2012



In this lab we will extend the image arithmetic class we developed in this week's lecture. You will overload the python operators that are used for bitwise logic (e.g. and, or, not, xor).

1. Create pydoc strings documenting all the methods.
2. Finish the class by implementing the `invert`, `or`, `xor`, and `gt` operators.
3. Experiment with image operators. How do they differ on color, grayscale, and binary images? Come up with some uses for the different operators.
4. Try out the following (mysterious?!) image expressions and comment on what function they perform.

```
ask("Press enter to take the background picture")
bg = BPicture(takePicture())
ask("Press enter to take the foreground picture")
fg = BPicture(takePicture())

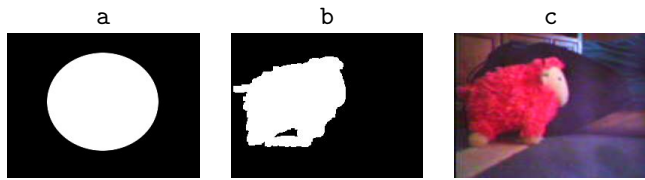
show(bg ^ fg)
show((bg ^ fg) > 20)
show(++-- ((bg ^ fg) > 20))
show(++-- ((bg ^ fg) > 20)) ^ + (++-- ((bg ^ fg) > 20))
show(~(+((bg ^ fg) > 20) >> 75) & bg)
show(~(+((bg ^ fg) > 20) >> 75) & bg | ((+((bg ^ fg) > 20) & fg) >> 75))
```

Learning Objectives

- Overload Operators
- Image Masks
- Binary Image Processing

Deliverables

Submit an electronic copy of your program and any supporting images via moodle as a zip file.



operator	method name	python operator	functional relationship	op(a, b)	op(a, c)
and	<code>--and--</code>	<code>a & b</code>	$\min(a, b)$		
or	<code>--or--</code>	<code>a b</code>	$\max(a, b)$		
xor	<code>--xor--</code>	<code>a ^ b</code>	$\text{abs}(a - b)$		
invert	<code>--invert--</code>	<code>~a</code>	$-a$		
threshold	<code>--gt--</code>	<code>c > 100</code>	$\text{binaryThreshold}(c, 100)$		
rshift	<code>--rshift--</code>	<code>c >> 50</code>	$\text{shift}(c, 50)$		
lshift	<code>--lshift--</code>	<code>c << 50</code>	$\text{shift}(c, -50)$		
dilate	<code>--pos--</code>	<code>+a</code>	$\text{dilate}(a, 3)$		
erode	<code>--neg--</code>	<code>-a</code>	$\text{erode}(a, 3)$		

```

from Myro import *
from Graphics import *

def brightness(px):
    ''' find the brightness of a pixel'''
    return sum(getRGB(px))/3

class BPicture(Picture):

    def __init__(self, p):
        Picture.__init__(self)

    def __and__(self, other):

        newImage = BPicture(copyPicture(self))

        for px in getPixels(newImage):
            x = getX(px)
            y = getY(px)
            opx = getPixel(other, x, y)
            setRed(px, min(getRed(px), getRed(opx)))
            setGreen(px, min(getGreen(px), getGreen(opx)))
            setBlue(px, min(getBlue(px), getBlue(opx)))

        return newImage

    def __rshift__(self, amount):

        newImage = BPicture(copyPicture(self))
        w = getWidth(newImage)

        for px in getPixels(newImage):
            x = getX(px)
            y = getY(px)
            if 0 < x - amount < w:
                setColor(px, getColor(self, x - amount, y))
            else:
                setColor(px, Color('black'))

        return newImage

    def __lshift__(self, amount):
        return self.__rshift__(-amount)

```

```

def __pos__(self):
    ''' dilate: expand the bright pixels'''
    w = getWidth(self)
    h = getHeight(self)
    newImage = BPicture(copyPicture(self))

    for px in getPixels(newImage):
        x = getX(px)
        y = getY(px)
        value = brightness(px)
        color = getColor(px)
        for dx in range(-1, 2, 1):
            for dy in range(-1, 2, 1):
                if 0 <= x + dx < w and 0 <= y + dy < h:
                    opx = getPixel(self, x + dx, y + dy)
                    if value < brightness(opx):
                        value = brightness(opx)
                        color = getColor(opx)
            setColor(px, color)

    return newImage

def __neg__(self):
    ''' erode: enlarge the dark pixels'''
    w = getWidth(self)
    h = getHeight(self)
    newImage = BPicture(copyPicture(self))

    for px in getPixels(newImage):
        x = getX(px)
        y = getY(px)
        value = brightness(px)
        color = getColor(px)
        for dx in range(-1, 2, 1):
            for dy in range(-1, 2, 1):
                if 0 <= x + dx < w and 0 <= y + dy < h:
                    opx = getPixel(self, x + dx, y + dy)
                    if value > brightness(opx):
                        value = brightness(opx)
                        color = getColor(opx)
            setColor(px, color)

    return newImage

```