**CMSC 143: Introduction to Object-Oriented Programming with Robots**

# Lab 11: Image Expressions

Due April 25, 2011

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 the methods already provided: `and`, `rshift`, `lshift`, `pos`, `neg`.
2. Finish the class by implementing the `or`, `xor`, `invert`, 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.

```
raw_input("Press enter to take the background picture")
bg = BPicture(takePicture())
raw_input("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))
```

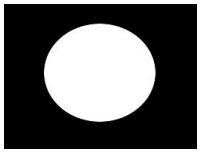
## Learning Objectives

◦ Overload Operators ◦ Image Masks ◦ Binary Image Processing

## Deliverables

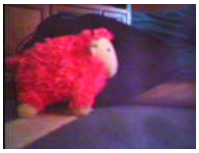
Submit an electronic copy of your program and any supporting images via moodle.

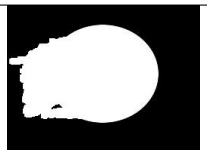
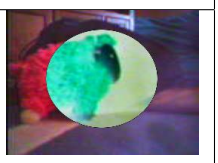
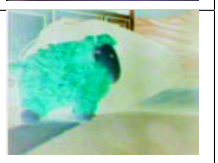
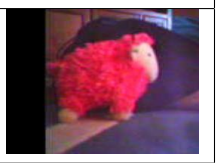
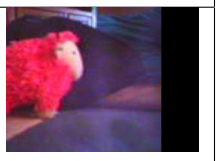
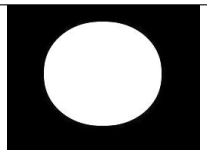
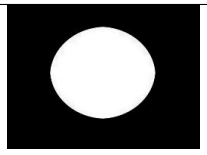
a



b

c



| operator | method name | python operator | functional relationship | op(a, b) | op(a, c) |
|---|---|---|---|---|---|
| and | `__and__` | `a & b` | $min(a, b)$ |  |  |
| or | `__or__` | `a \| b` | $max(a, b)$ |  |  |
| xor | `__xor__` | `a ^ b` | $abs(a - b)$ |  |  |
| invert | `__invert__` | `~a` | $-a$ |  |  |
| threshold | `__gt__` | `c > 100` | $binaryThreshold(c, 100)$ | |  |
| rshift | `__rshift__` | `c >> 50` | $shift(c, 50)$ | |  |
| lshift | `__lshift__` | `c << 50` | $shift(c, -50)$ | |  |
| dilate | `__pos__` | `+a` | $dilate(a, 3)$ |  | |
| erode | `__neg__` | `-a` | $erode(a, 3)$ |  | |

```
from myro 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)
        self.set(p.getWidth(), p.getHeight(), p.image, "image")

    def __and__(self, other):

        newImage = BPicture(makePicture(self.getWidth(), self.getHeight()))

        for newPixel in getPixels(newImage):
            x = getX(newPixel)
            y = getY(newPixel)
            spx = getPixel(self, x, y)
            opx = getPixel(other, x, y)
            setRed(newPixel, min(getRed(spx), getRed(opx)))
            setGreen(newPixel, min(getGreen(spx), getGreen(opx)))
            setBlue(newPixel, min(getBlue(spx), 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, getPixel(self, x - amount, y))
            else:
                setColor(px, 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
```