CS 1124
Media Computation

Steve Harrison

Lecture 4.1 (September 15, 2008)

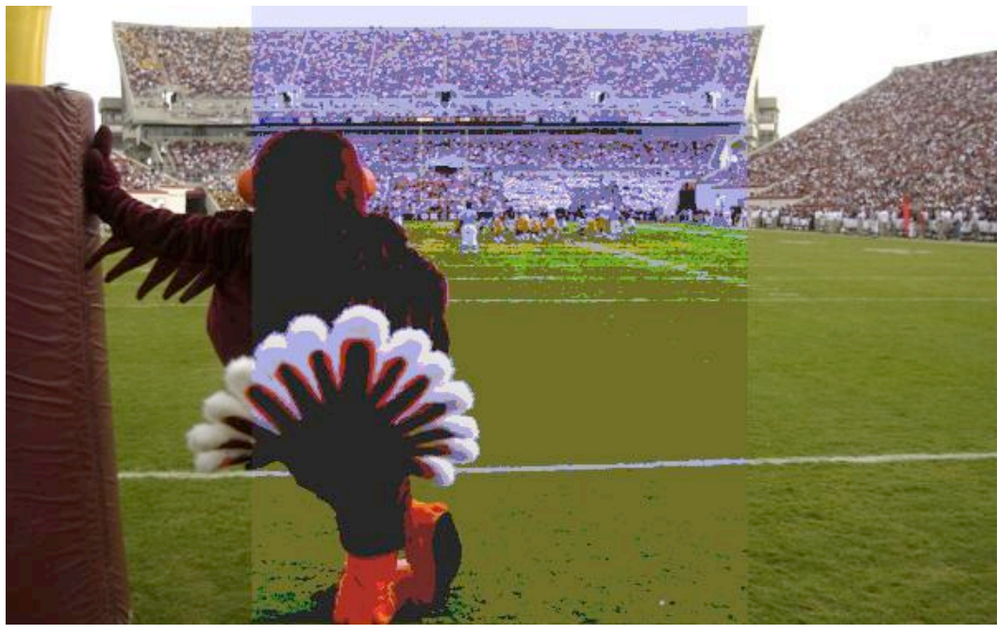
Today ...

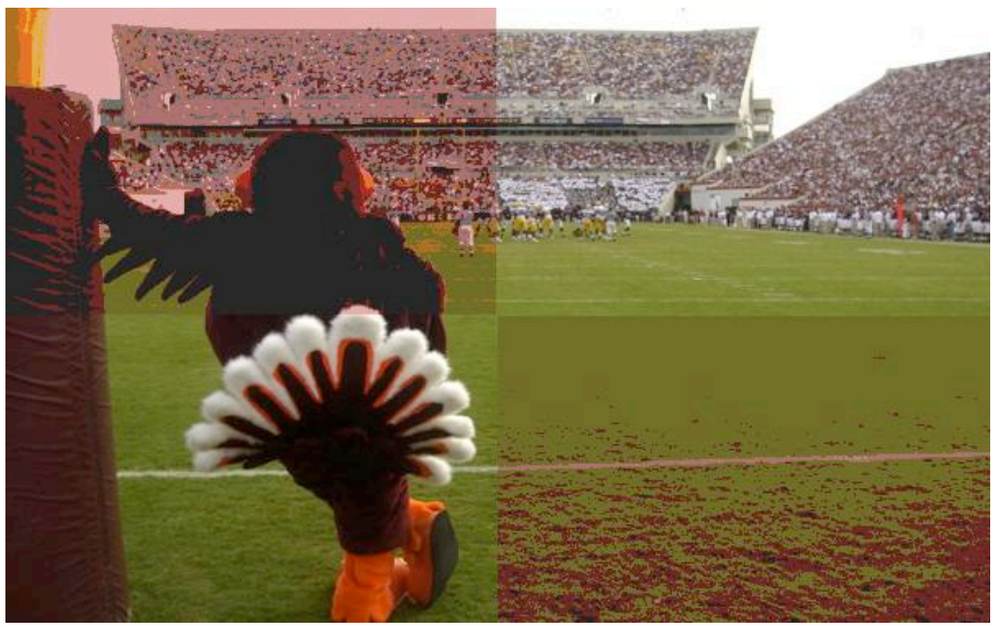
- HW 2 review
  - **cool lagniappes**
  - **a few problems to work on**
- Copying
  - **we left off trying to write a scale-up function....**
  - **lets back up and review some stuff**
  - **some heuristics about copying, placing images in other images, and scaling**

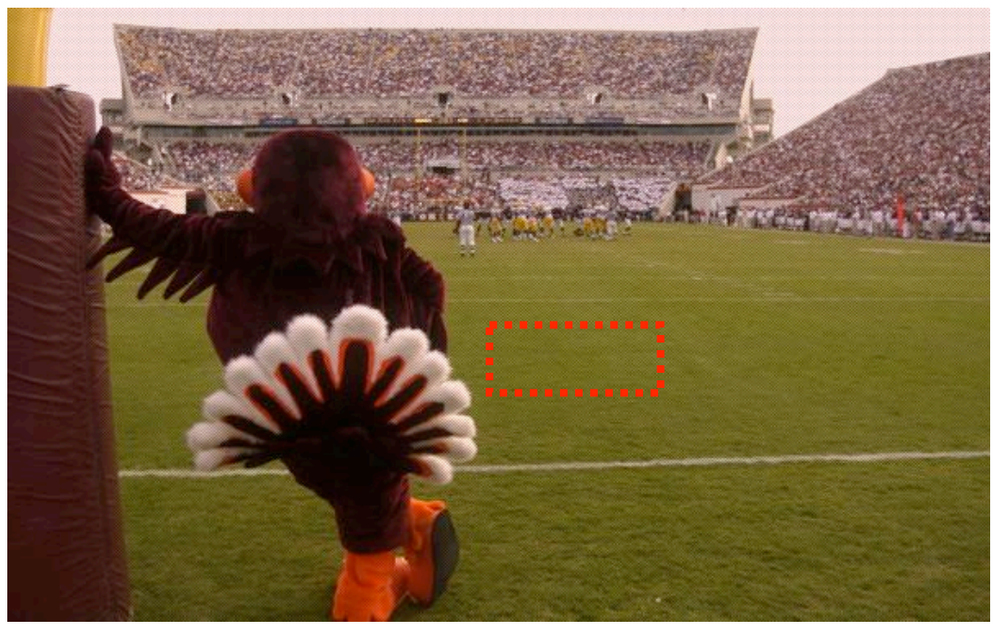
Most did very well !

Points	Number
100	12
> 94	9
90 - 95	4
< 90	6

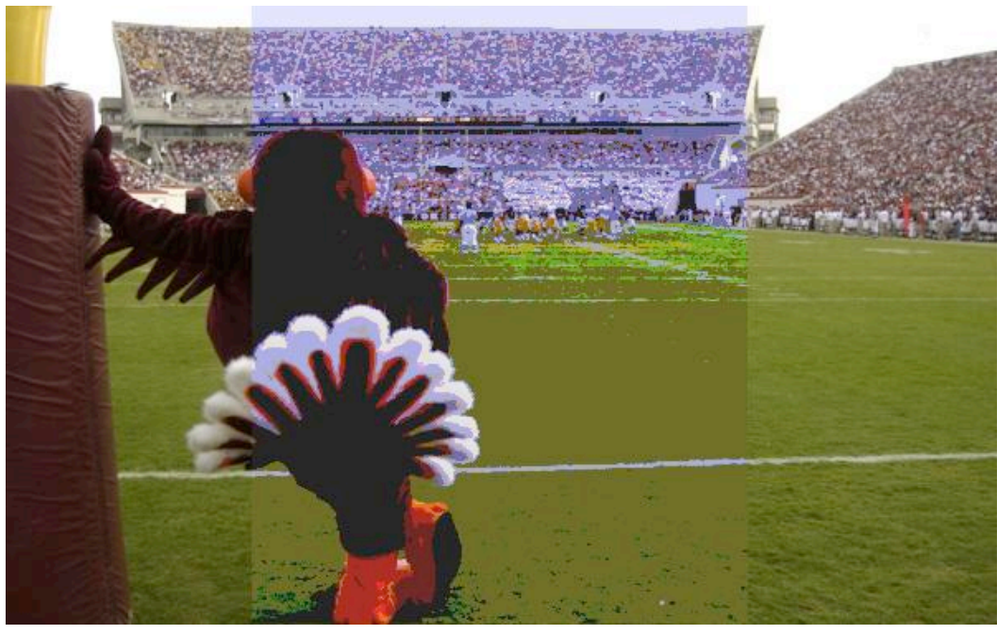
Cool lagniappes

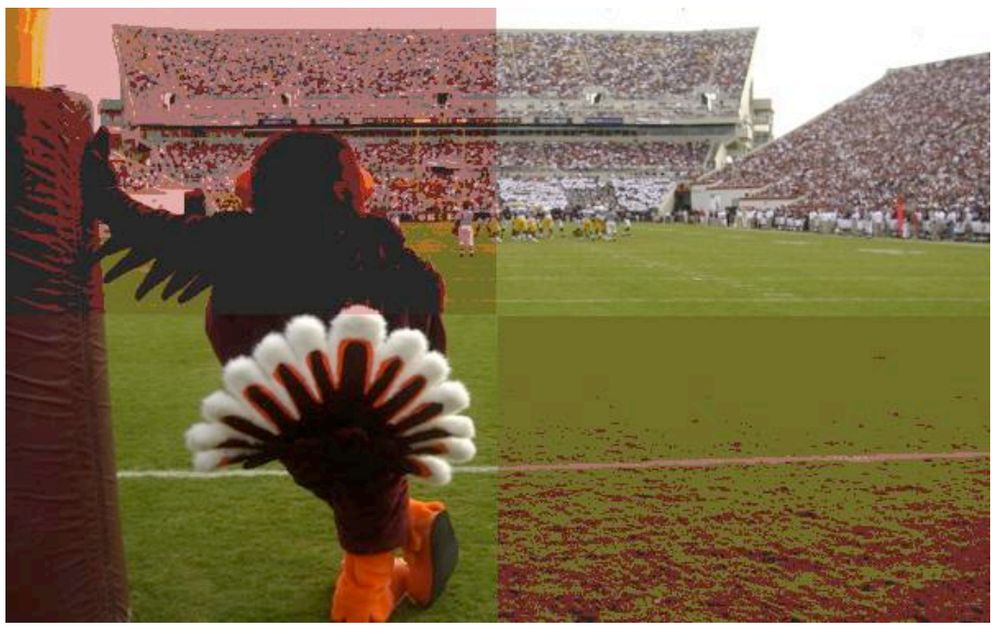


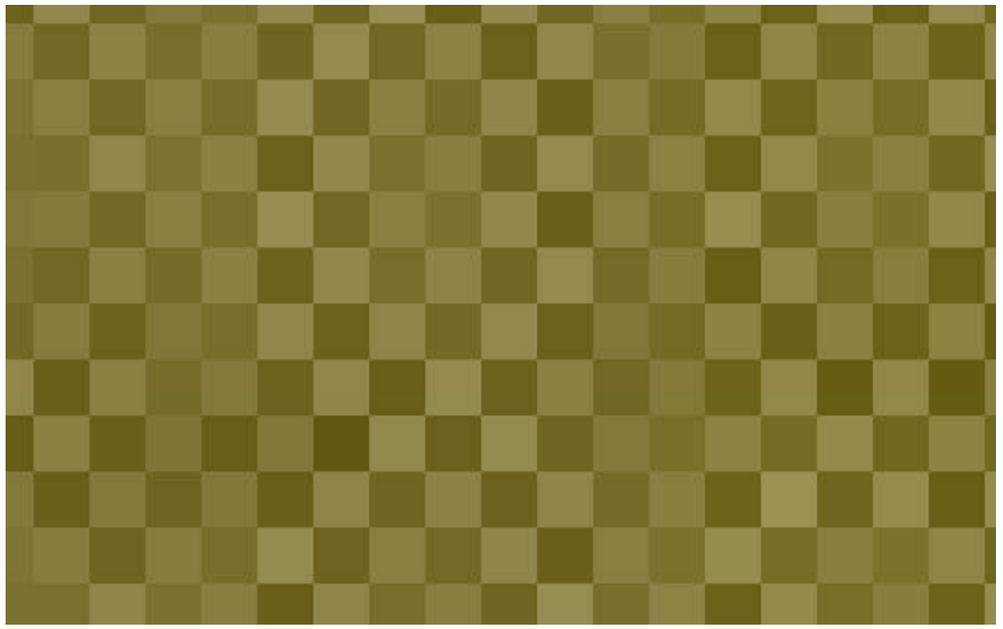




Cool lagniappes







Problems seen in HW 2

- Not decomposed into functions
- Not reading CLOSELY the assignment

Some really clever recipes

- A couple observed that pixel-level module for posterize was better than picture-level

```
def poster( pixel ) :
```

- Some passed color-changing factor

```
def decreaseRed( picture, amount ) :
```

- One noticed that combining functions needed flag to trigger writing/not-writing file

```
def decreaseRed( picture, amount, writeFlagTF ) :
```

I don't think anyone noticed ...

That you only had to create FIVE new pictures using makePicture(file) since the sixth picture was to made by combining two of the functions:

```
def makePictures( file ) :
  redPic = makePicture(file)
  greenPic = makePicture(file)
  bluePic = makePicture(file)
  sunsetPic=makePicture(file)
  postrPic = makePicture(file)

  redPic = makeRedPic( redPic )
  bluePic = makeBluePic( bluePic )
  greenPic = makeGreenPic( greenPic )
  sunsetPic=makeSunsetPic( sunsetPic )
  postrPic=makePostrPic( posterPic )
  comboPic = makeComboPic( sunsetPic )


def comboPic( picture ) :
  return makePostrPic( picture )

def makePostrPic( picture ) :
.
.
.
def makeRed( picture ) :
.
.
.
.
```

I don't think anyone noticed ...

That you only had to create FIVE new pictures using makePicture(file) since the sixth picture was to made by combining two of the functions:

```
def makePictures( file ) :
  redPic = makePicture(file)
  greenPic = makePicture(file)
  bluePic = makePicture(file)
  sunsetPic=makePicture(file)
  postrPic = makePicture(file)

  redPic = makeRedPic( redPic )
  bluePic = makeBluePic( bluePic )
  greenPic = makeGreenPic( greenPic )
  sunsetPic=makeSunsetPic( sunsetPic )
  postrPic=makePostrPic( posterPic )
  comboPic = makePostrPic( sunsetPic )
```

I don't think anyone noticed ...

You could write one change color function and pass in the amount to change each. This would be slower for changeRed, changeBlue, changeGreen, but faster for sunset and maybe combo.

```
def changeColor( picture, redAmount, greenAMount, blueAmount ) :

    for pxl in getPixels( picture ) :
      pxlRed = int( getRed( pxl )* redAmount )
      pxlGreen = int( getGreen( pxl ) * greenAmount )
      pxlBlue = int( getBlue( pxlBlue ) * blueAmount )
      setColor( pxl, pxlRed, pxlGreen, pxlBlue )
```

Today ...

- HW 2 review
  - **cool lagniappes**
  - **a few problems to work on**
- Copying
  - **we left off trying to write a scale-up function....**
  - **lets back up and review some stuff**
  - **some heuristics about copying, placing images in other images, and scaling**

Scaling

- Scaling a picture (smaller or larger) has to do with *sampling* the source picture differently
  - **When we just copy, we sample every pixel**
  - **If we want a smaller copy, we skip some pixels**
    - We *sample* fewer pixels
  - **If we want a larger copy, we duplicate some pixels**
    - We *over-sample* some pixels

Scaling the picture down

```
def copyPictureHalfAsBig( file ):
  # Set up the source and target pictures
  pic = makePicture(file)
  canvasFile = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasFile)
  # Now, do the actual copying
  sourceX = 45
  for targetX in range(100,100+((200-45)/2)):
    sourceY = 25
    for targetY in range(100,100+((200-25)/2)):
      color = getColor(getPixel(pic,sourceX,sourceY))
      setColor(getPixel(canvas,targetX,targetY), color)
      sourceY = sourceY + 2
    sourceX = sourceX + 2
  show(pic)
  show(canvas)
  return canvas
```

```
>>> barbFile = pickAFile()
>>> setMediaPath()
>>> smallPic = copyPictureHalfAsBig( barbFile )
```

Blank files in mediasources

- getMediaPath(“7inX95in.jpg”) gives you a JPEG canvas which prints out as 7x9.5 inches
  - **Letter-sized page with 1 inch margins**
- getMediaPath(“640x480.jpg”) gives a JPEG canvas at a common size: 640 pixels across by 480 pixels high

Scaling Up: Growing the picture

- First - a reminder about integer and real numbers

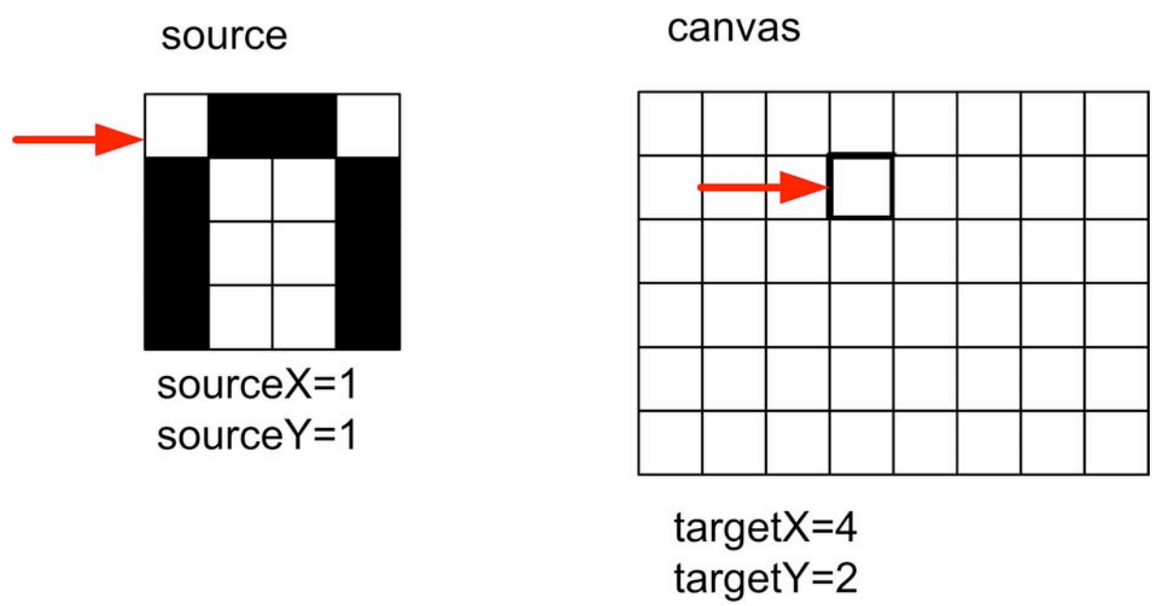
Scaling Up: Growing the picture

- To grow a picture, we simply duplicate some pixels
- We do this by incrementing by 0.5, but only use the integer part
- (Remember our x & y's must be integer)

```
>>> print int(1)
1
>>> print int(1.5)
1
>>> print int(2)
2
>>> print int(2.5)
2
```

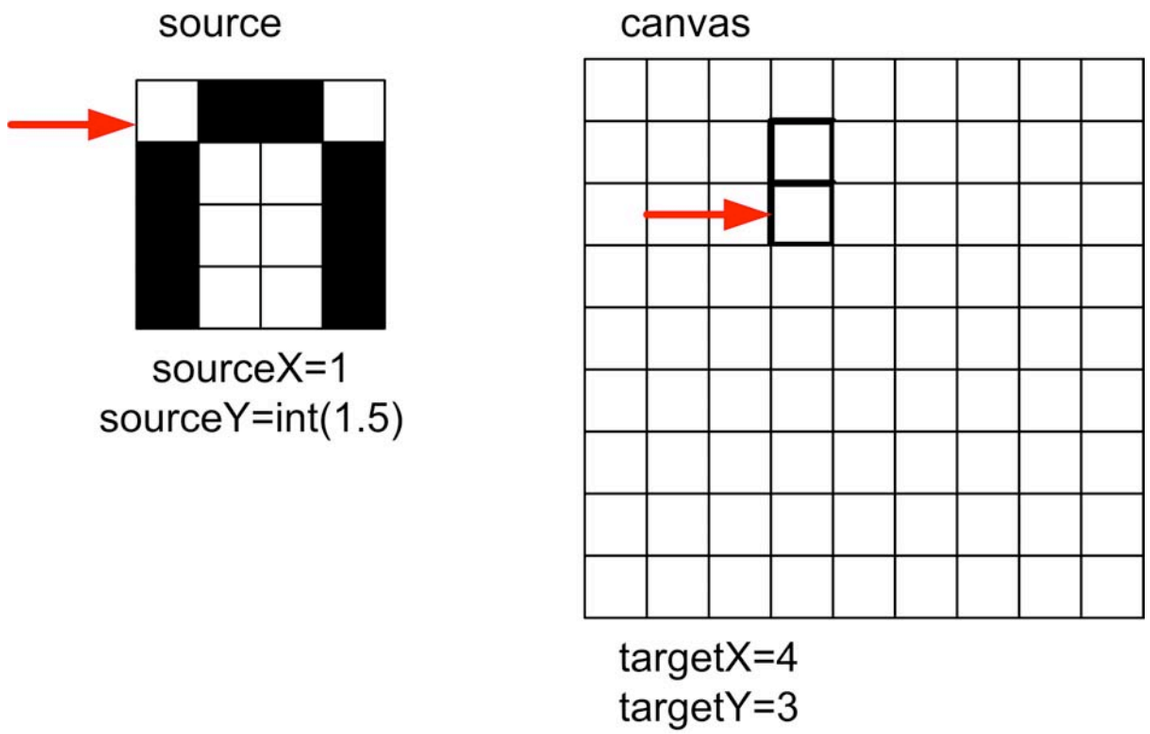
Scaling up: How it works

- Same basic setup as copying and rotating:



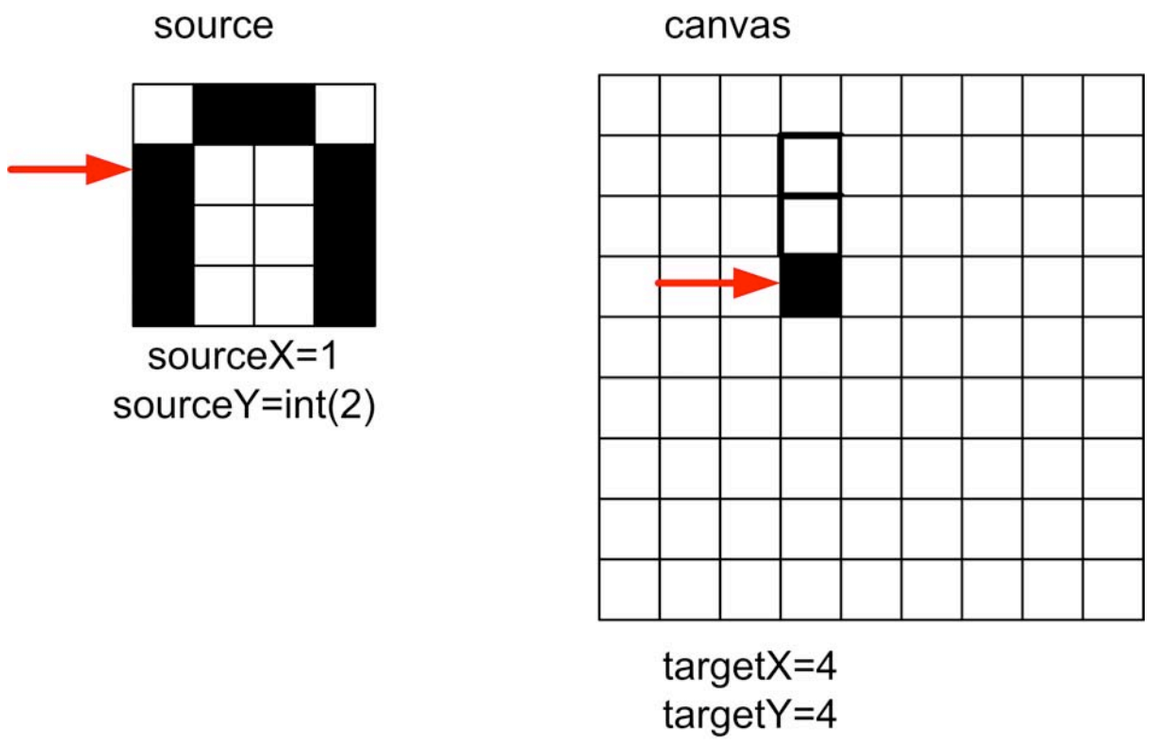
Scaling up: How it works 2

- But as we increment by *only 0.5*, and we use the **int()** function, we end up taking every pixel *twice*.
- Here, the blank pixel at (1,1) in the source gets copied twice onto the canvas.



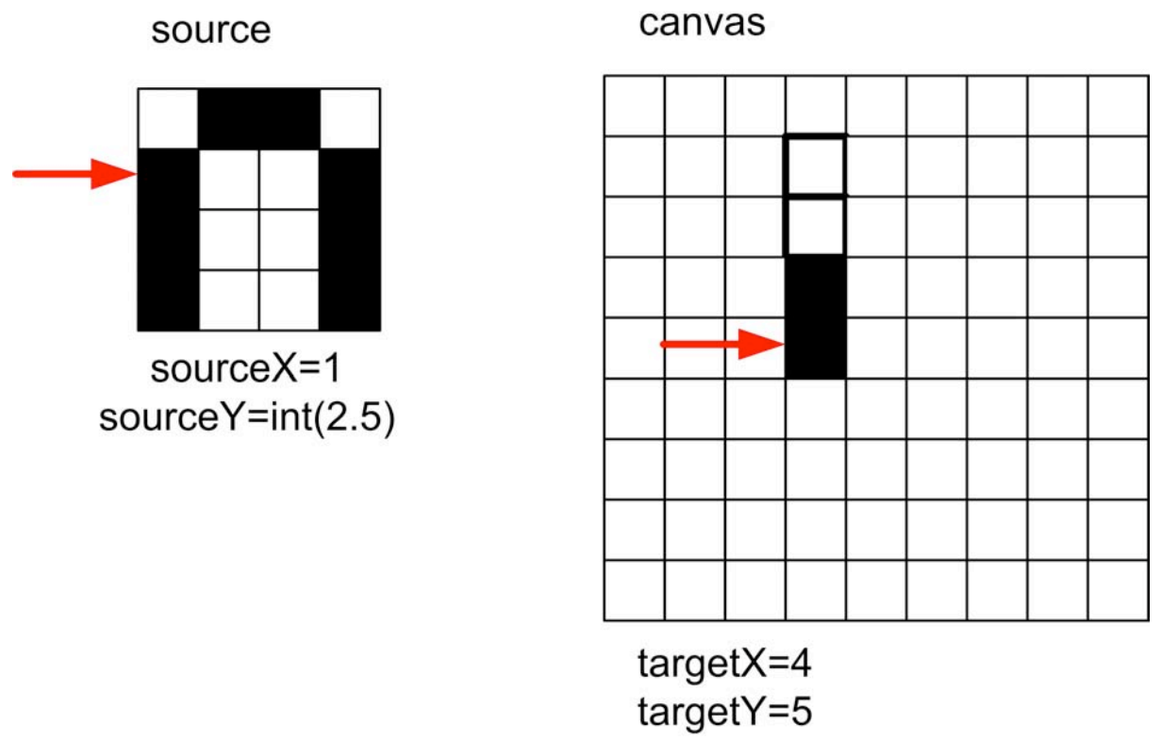
Scaling up: How it works 3

- Black pixels gets copied once…



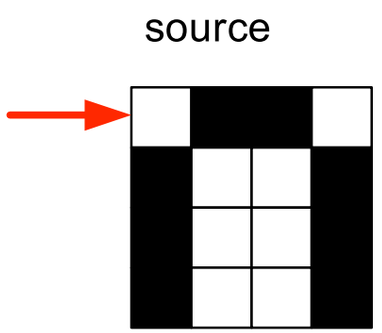
Scaling up: How it works 4

- And twice…

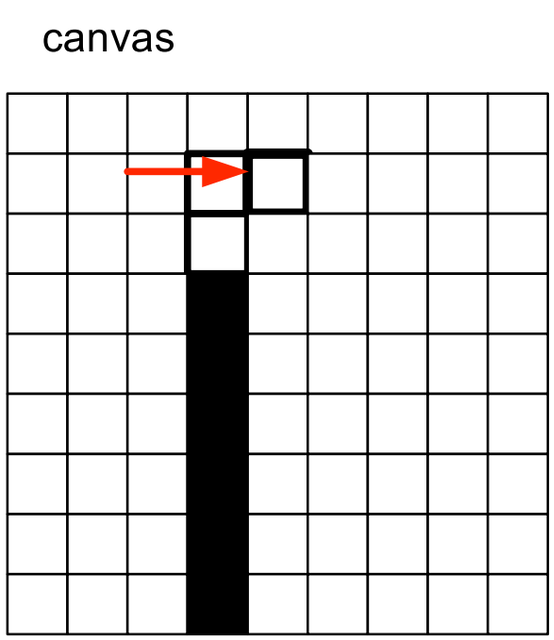


Scaling up: How it works 5

- The next “column” (x) in the source, is the *same* “column” (x) in the target.



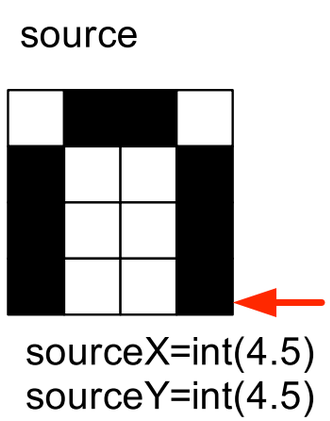
sourceX=int(1.5)
sourceY=int(1)

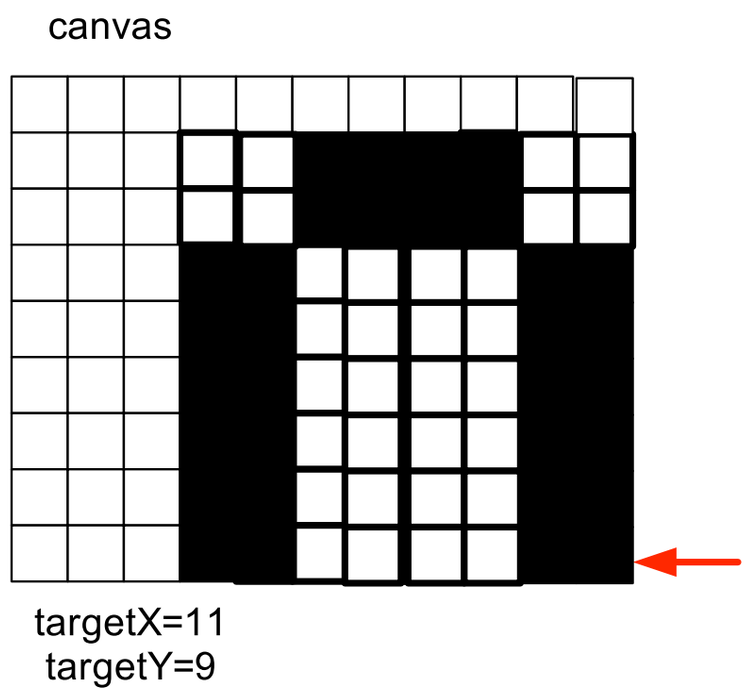


targetX=5
targetY=2

Scaling up: How it ends up

- We end up in the same place in the source, but twice as much in the target.
- Notice the degradation:
  - **Curves get “choppy”: Pixelated**





Copying pixels

Copying pixels

- In general, what we want to do is to keep track of a sourceX and sourceY, and a targetX and targetY.
  - **We increment (add to them) in pairs**
    - sourceX and targetX get incremented together
    - sourceY and targetY get incremented together
  - **The tricky parts are:**
    - Setting values *inside* the body of loops
    - Incrementing at the *bottom* of loops

Lets figure out how to make a copy

- copyBarb()

Copying Barb to a canvas

```
def copyBarb():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  targetX = 1
  for sourceX in range(1,getWidth(barb)+1):
    targetY = 1
    for sourceY in range(1,getHeight(barb)+1):
      color = getColor(getPixel(barb,sourceX,sourceY))
      setColor(getPixel(canvas,targetX,targetY), color)
      targetY = targetY + 1
    targetX = targetX + 1
  show(barb)
  show(canvas)
  return canvas
```

Walking through the copying function

- First, get the source (barb) and target (canvas) files and pictures as names we can use later.

```
def copyBarb():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
 # Now, do the actual copying
targetX = 1
for sourceX in range(1,getWidth(barb)+1):
  targetY = 1
  for sourceY in range(1,getHeight(barb)+1):
    color = getColor(getPixel(barb,sourceX,sourceY))
    setColor(getPixel(canvas,targetX,targetY), color)
    targetY = targetY + 1
  targetX = targetX + 1
show(barb)
```

The actual copy

- We get the color of the pixel at sourceX and sourceY
- We set (copy) the color to the pixel in the target picture at targetX and targetY

```
def copyBarb():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  targetX = 1
  for sourceX in range(1,getWidth(ba
    targetY = 1
    for sourceY in range(1,getHeight(
      color = getColor(getPixel(barb,s
      setColor(getPixel(canvas,target
      targetY = targetY + 1
    targetX = targetX + 1
  show(barb)
  show(canvas)
  return canvas
```

The actual copy

- We get the color of the pixel at sourceX and sourceY
- We set (copy) the color to the pixel in the target picture at targetX and targetY

```
targetX = 1
for sourceX in range(1,getWidth(barb)+1):
  targetY = 1
  for sourceY in range(1,getHeight(barb)+1):
    color = getColor(getPixel(barb,sourceX,sourceY))
    setColor(getPixel(canvas,targetX,targetY), color)
    targetY = targetY + 1
  targetX = targetX + 1
```

Setting up the copy loop

- targetX gets set to 1 at the beginning
- sourceX will range across the width of the source picture
- *INSIDE* the loop, we set targetY to 1
  - **Inside because we want it to start at 1 each time we do a new X**
- sourceY will range from 1 to height of source

```
def copyBarb():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  targetX = 1
  for sourceX in range(1,getWidth(barb)+1):
    targetY = 1
    for sourceY in range(1,getHeight(barb)+1):
      color = getColor(getPixel(barb,sourceX,sourc
      setColor(getPixel(canvas,targetX,targetY), co
      targetY = targetY + 1
    targetX = targetX + 1
  show(barb)
  show(canvas)
  return canvas
```

Ending the loop

- Just before we end the sourceY loop, we increment targetY
  - **It's now set up for the next time through the loop**
  - **It's set correctly for the next value of sourceY**
- Just before we end the sourceX loop, we increment the targetX
  - **Note carefully the indentation to figure out which goes with which loop**

```
def copyBarb():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  targetX = 1
  for sourceX in range(1,getWidth(barb)+1):
    targetY = 1
    for sourceY in range(1,getHeight(barb)+1):
      color = getColor(getPixel(barb,sourceX,sourceY))
      setColor(getPixel(canvas,targetX,targetY), color)
      targetY = targetY + 1
    targetX = targetX + 1
  show(barb)
  show(canvas)
  return canvas
```

Ending the copy function

- At the very end, we show the source and target
- And return the modified target.
- Why do we need the return?

```
def copyBarb():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  targetX = 1
 for sourceX in range(1,getWidth(barb)+1):
   targetY = 1
   for sourceY in range(1,getHeight(barb)+1):
     color = getColor(getPixel(barb,sourceX,sourceY))
     setColor(getPixel(canvas,targetX,targetY), color)
     targetY = targetY + 1
   targetX = targetX + 1
 show(barb)
 show(canvas)
 return canvas
```

Works either way

```
def copyBarb2():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  sourceX = 1
  for targetX in range(1,getWidth(barb)+1):
    sourceY = 1
    for targetY in range(1,getHeight(barb)+1):
      color = getColor(getPixel(barb,sourceX,sourceY))
      setColor(getPixel(canvas,targetX,targetY), color)
      sourceY = sourceY + 1
    sourceX = sourceX + 1
  show(barb)
  show(canvas)
  return canvas
```

As long as we increment sourceX and targetX together, and sourceY and targetY together, it doesn't matter which is in the for **loop and which is incremented via expression**

Transformation =

Small changes in copying

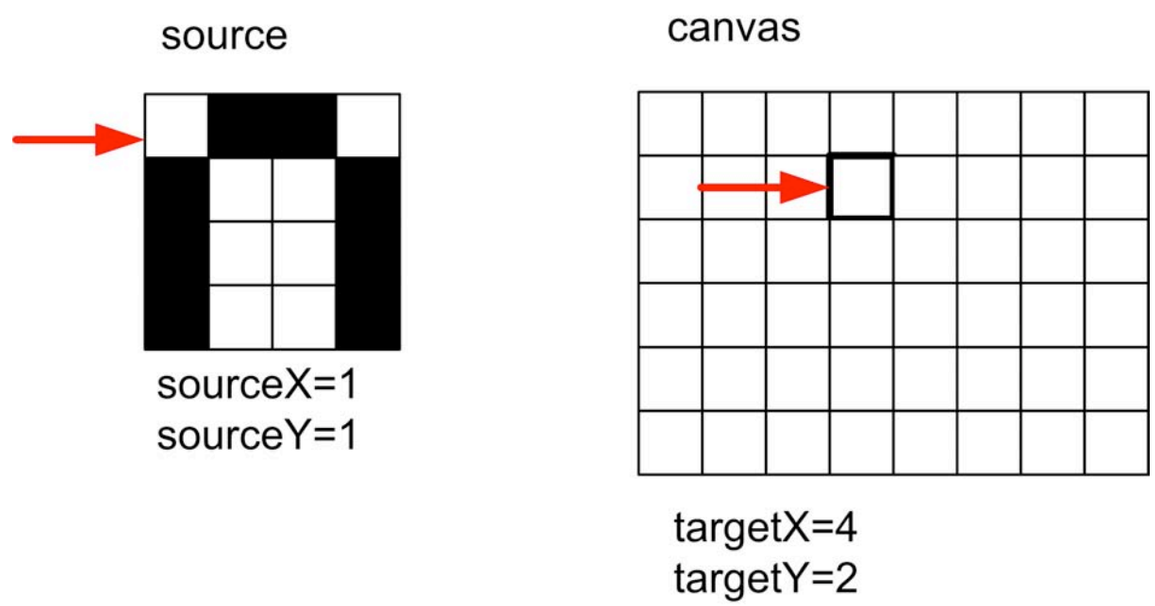
- Making relatively small changes in this basic copying program can make a variety of transformations. These are ***heuristics***.
  - **Change the targetX and targetY, and you copy to wherever you want**
  - **Cropping: Change the sourceX and sourceY range, and you copy only part of the image.**
  - **Rotating: Swap targetX and targetY, and you end up copying sideways**
  - **Scaling: Change the increment on sourceX and sourceY, and you either enlarge or shrink the image.**

Copying into the middle of the canvas

```
def copyBarbMidway():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  targetX = 100
  for sourceX in range(1,getWidth(barb)+1):
    targetY = 100
    for sourceY in range(1,getHeight(barb)+1):
      color = getColor(getPixel(barb,sourceX,sourceY))
      setColor(getPixel(canvas,targetX,targetY), color)
      targetY = targetY + 1
    targetX = targetX + 1
  show(barb)
  show(canvas)
  return canvas
```

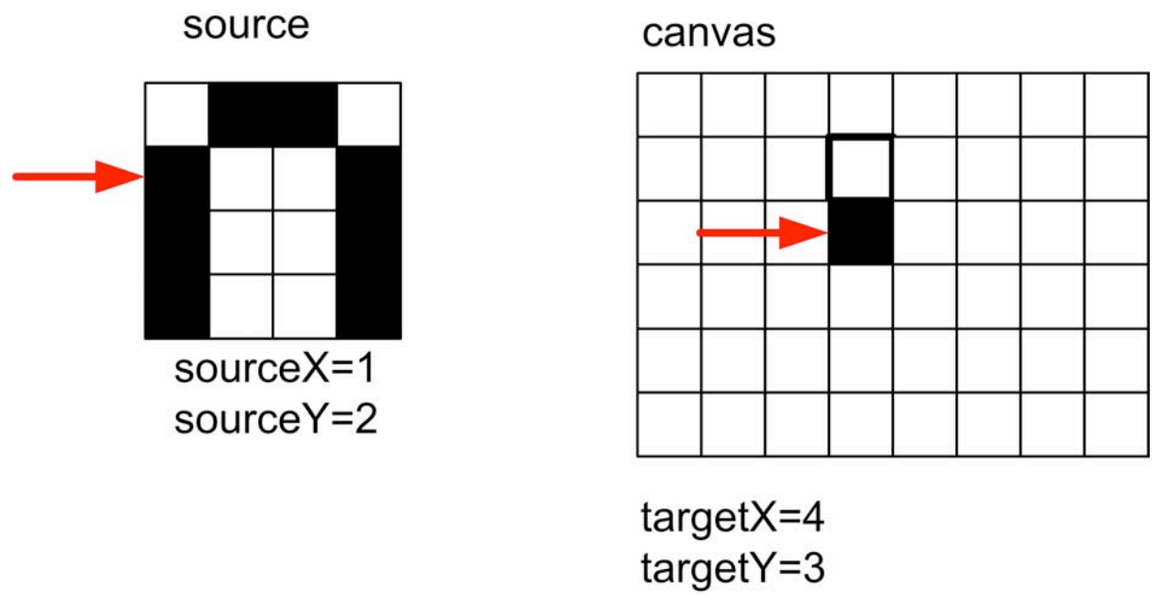
Copying: How it works

- Here's the initial setup:



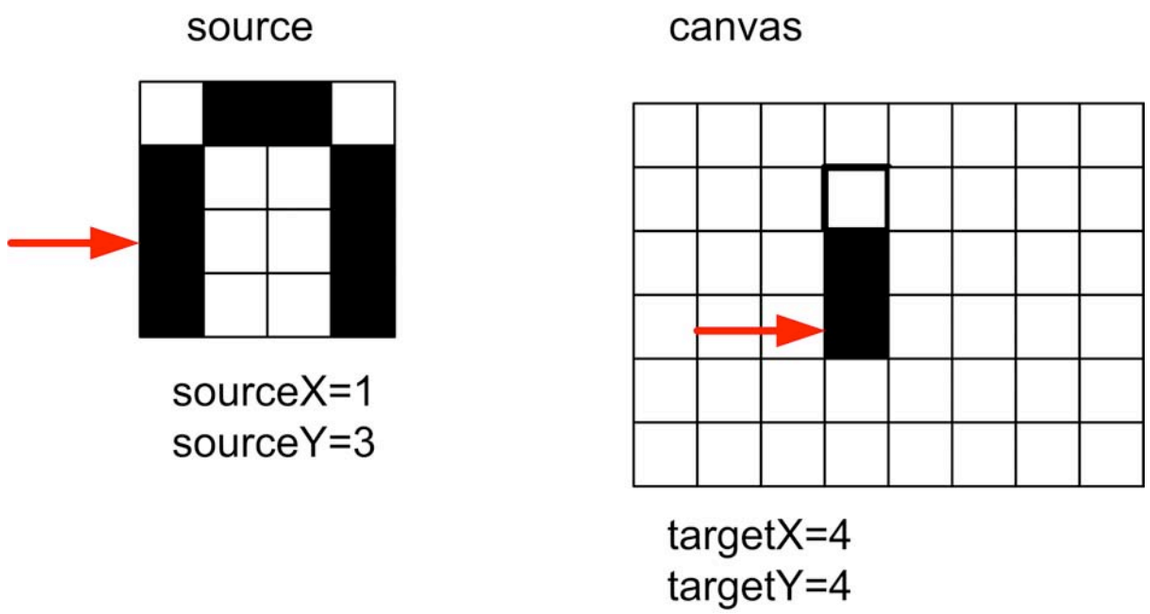
Copying: How it works 2

- After incrementing the sourceY and targetY once (whether in the **for** or via expression):



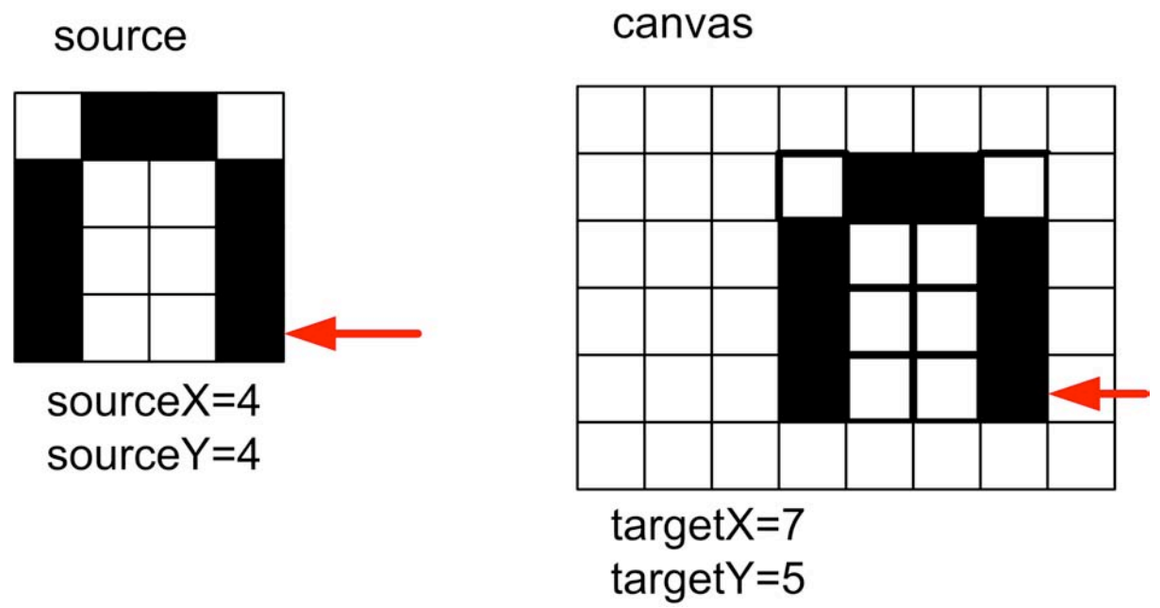
Copying: How it works 3

- After yet another increment of sourceY and targetY:
- When we finish that column, we increment sourceX and targetX, and start on the next column.



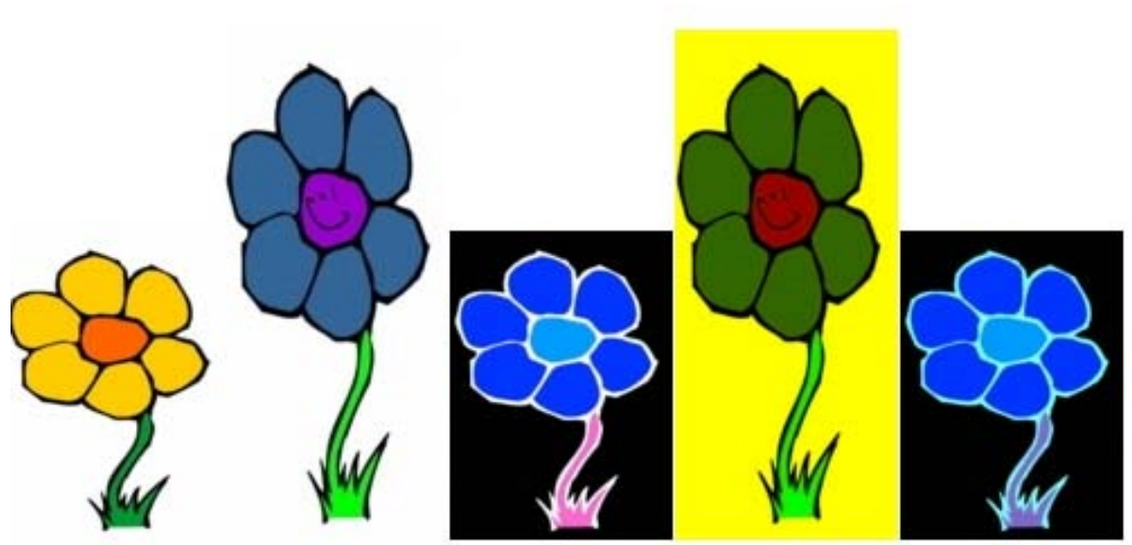
Copying: How it looks at the end

- Eventually, we copy every pixel



Making a collage

- Could we do something to the pictures we copy in?
  - **Sure! Could either apply one of those functions before copying, or do something to the pixels during the copy.**
- Could we copy more than one picture!
  - **Of course! Make a collage!**



```
def createCollage():
  flower1=makePicture(getMediaPath("flower1.jpg"))
  print flower1
  flower2=makePicture(getMediaPath("flower2.jpg"))
  print flower2
  canvas=makePicture(getMediaPath("640x480.jpg"))
  print canvas
  #First picture, at left edge
  targetX=1
  for sourceX in range(1,getWidth(flower1)):
    targetY=getHeight(canvas)-getHeight(flower1)-5
    for sourceY in range(1,getHeight(flower1)):
      px=getPixel(flower1,sourceX,sourceY)
      cx=getPixel(canvas,targetX,targetY)
      setColor(cx,getColor(px))
      targetY=targetY + 1
    targetX=targetX + 1
  #Second picture, 100 pixels over
  targetX=100
  for sourceX in range(1,getWidth(flower2)):
    targetY=getHeight(canvas)-getHeight(flower2)-5
    for sourceY in range(1,getHeight(flower2)):
      px=getPixel(flower2,sourceX,sourceY)
      cx=getPixel(canvas,targetX,targetY)
      setColor(cx,getColor(px))
      targetY=targetY + 1
    targetX=targetX + 1
```

Exactly from book

```
  #Third picture, flower1 negated
  negative(flower1)
  targetX=200
  for sourceX in range(1,getWidth(flower1)):
    targetY=getHeight(canvas)-getHeight(flower1)-5
    for sourceY in range(1,getHeight(flower1)):
      px=getPixel(flower1,sourceX,sourceY)
      cx=getPixel(canvas,targetX,targetY)
      setColor(cx,getColor(px))
      targetY=targetY + 1
    targetX=targetX + 1
  #Fourth picture, flower2 with no blue
  clearBlue(flower2)
  targetX=300
  for sourceX in range(1,getWidth(flower2)):
    targetY=getHeight(canvas)-getHeight(flower2)-5
    for sourceY in range(1,getHeight(flower2)):
      px=getPixel(flower2,sourceX,sourceY)
      cx=getPixel(canvas,targetX,targetY)
      setColor(cx,getColor(px))
      targetY=targetY + 1
    targetX=targetX + 1
  #Fifth picture, flower1, negated with decreased red
  decreaseRed(flower1)
  targetX=400
  for sourceX in range(1,getWidth(flower1)):
    targetY=getHeight(canvas)-getHeight(flower1)-5
    for sourceY in range(1,getHeight(flower1)):
      px=getPixel(flower1,sourceX,sourceY)
      cx=getPixel(canvas,targetX,targetY)
      setColor(cx,getColor(px))
      targetY=targetY + 1
    targetX=targetX + 1
  show(canvas)
  return(canvas)
```

What a Mess! How would you clean up createCollage() ?

- Why does targetY always start at **getHeight(canvas)-getHeight(flower2)-5** ?
- Notice that a lot of code repeats, can it be modularized?
- Notice also that it makes 5 different pictures from only 2 originals -- and that number five depends on having made flower1 a negative already

Cropping: Just the face

```
def copyBarbsFace():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  targetX = 100
  for sourceX in range(45,200):
    targetY = 100
    for sourceY in range(25,200):
      color = getColor(getPixel(barb,sourceX,sourceY))
      setColor(getPixel(canvas,targetX,targetY), color)
      targetY = targetY + 1
    targetX = targetX + 1
  show(barb)
  show(canvas)
  return canvas
```

Again, swapping the loop works fine

```
def copyBarbsFace2():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
# Now, do the actual copying
sourceX = 45
for targetX in range(100,100+(200-45)):
  sourceY = 25
  for targetY in range(100,100+(200-25)):
    color = getColor(getPixel(barb,sourceX,sourceY))
    setColor(getPixel(canvas,targetX,targetY), color)
    sourceY = sourceY + 1
  sourceX = sourceX + 1
show(barb)
show(canvas)
return canvas
```

We can use targetX and targetY as the **for** loop index variables, and everything works the same.

In HW 3 (signs in pictures)

- Which is source and which is target?

HW 3 HINTS

- Incrementing by a *real* number (e.g. 0.9 or 1.5) may mean that you need to change an integer to a real.
  - **to change make a real number out of an integer, multiply by 1.0**

```
targetRealX = targetX * 1.0
```

  - **But if you do, be sure to convert to an integer when getting or writing to a pixel**

```
pxl = getPixel( picture, int( targetRealX ), targetY )
```

- Putting the sign into the space for the sign may require some scaling in x and/or in y

```
sourceXRealStep = getWidth(source)*1.0/getWidth(target)*1.0
```

Today ...

- HW 2 review
  - **cool lagniappes**
  - **a few problems to work on**
- Copying
  - **we left off trying to write a scale-up function....**
  - **lets back up and review some stuff**
  - **some heuristics about copying, placing images in other images, and scaling**

Coming Attractions

- Wednesday
  - **contact your group members to decide which of your three options you will create**
- Friday
  - **Assignment 3 Due 2:00 PM**
- Next Monday
  - **Read Chapter 5**
  - **Quiz 5 due 10:00 AM**
- next Friday (9/26)
  -