CS 1124
Media Computation

Steve Harrison

Lecture 3.3 lab (September 12, 2008)

Today ...

- class project
  - **Look up some stuff about “abstraction”**
  - **break into groups**
- Using ranges in pictures
  - **scaling**

Abstraction

- Any reports?

Abstraction

Abstraction is the process or result of generalization by reducing the information content of a concept or an observable phenomenon, typically in order to retain only information which is relevant for a particular purpose.

Abstraction

- In computer science:
  - Computer scientists use abstraction to understand and solve problems and communicate their solutions with the computer in some particular computer language.
- In art
- In philosophy
  - Abstracting a leather soccer ball to a ball retains only the information on general ball attributes and behaviour. Similarly, abstracting happiness to an emotional state reduces the amount of information conveyed about the emotional state.

Group Project 1: visual abstraction

- Break into groups
- Come up with a project
  - **each student will find ONE picture**
  - **your group will come up with a recipe that creates a single abstraction from all (four or) five of your group's pictures**
- In next 30 minutes, come up with 3 alternatives
- Over the next week: research, discuss and choose one alternative to write.
- Also for next Friday, find and bring pictures to lab.

Group Project 1: visual abstraction

- Break into groups

1	2	3	4	5	6	7
Burke	D'Augustine	Maier	Combs	Currin	Malhotra	Bowers
Burton	Demase	Regione	Duffy	Dahiya	Rhyner	Davies
Howell	Ha	Taylor	Highman	Hughes	Roithmayr	Ho
Nassery	Heitzer	Thayer	Talley	Knowles	Slack	Pham
Zhang	Messick	Walsh	Tran	Merrow	Ota	

Today ...

- class project
  - **Look up some stuff about “abstraction”**
  - **break into groups**
- Using ranges in pictures
  - **scaling**

Moving pixels across pictures

- We've seen using index variables to track the pixel position we're working with in a picture.
- We can copy *between* pictures, if we keep track of:
  - **The source index variables**
    - Where we're getting the pixels *from*
  - **The target index variables**
    - Where we're putting the pixels *at*
- (Not really copying the pixels: Replicating their colors.)

What can you do then?

- What can you do when copying from one picture to another?
  - **Collages: Copy several pictures onto one**
  - **Cropping: You don’t have to take the whole picture**
  - **Scaling: Make a picture smaller, or larger when copying it**

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 )
```

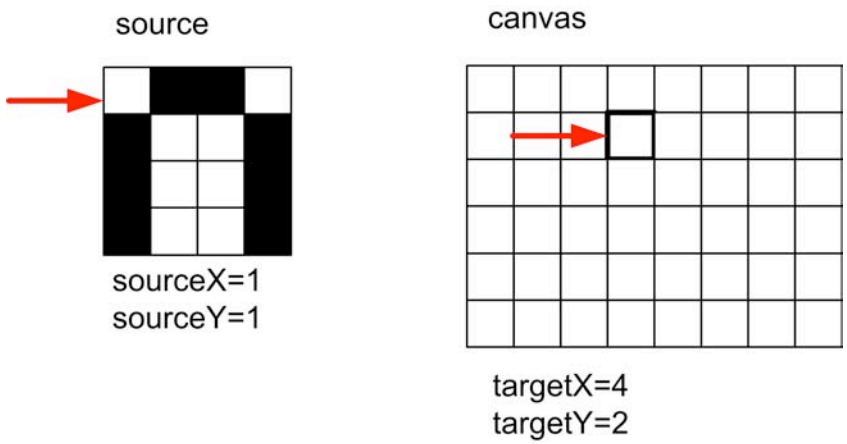
Scaling Up: Growing the picture

- 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
```

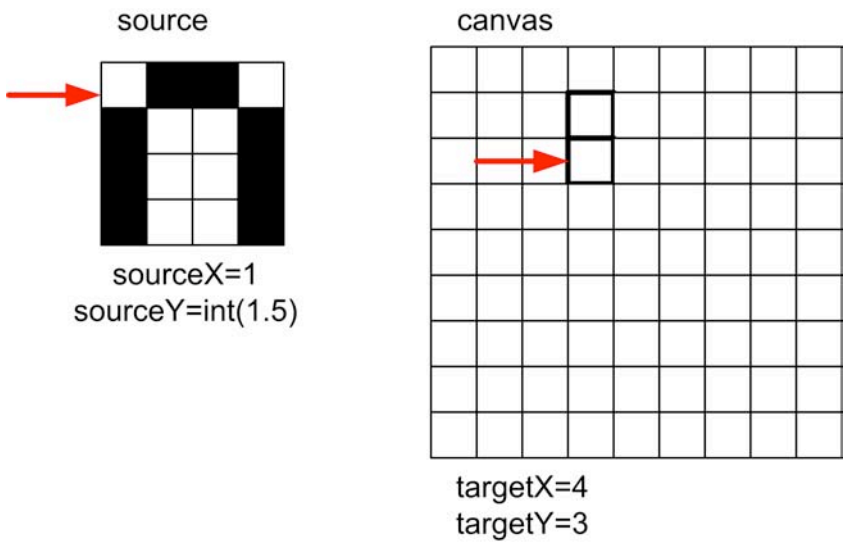
Scaling up: How it works

- Same basic setup as copying and rotating:



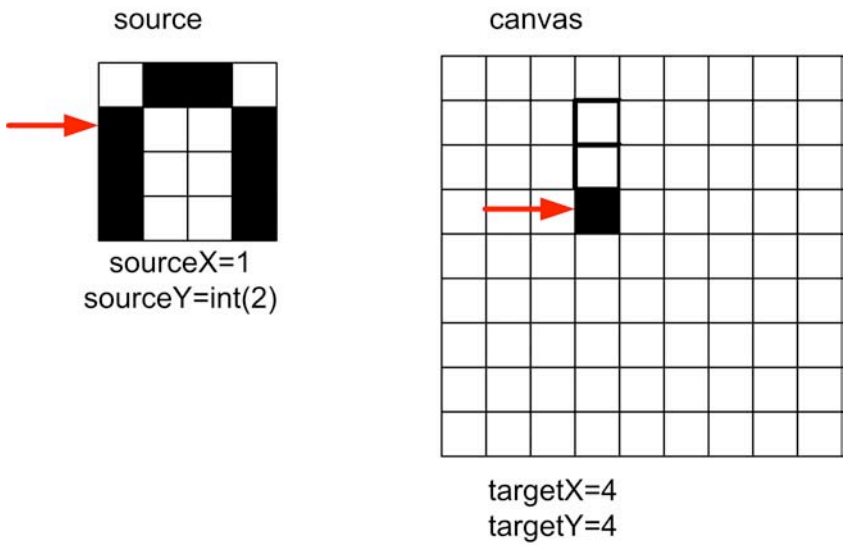
Scaling up: How it works 2

- 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.



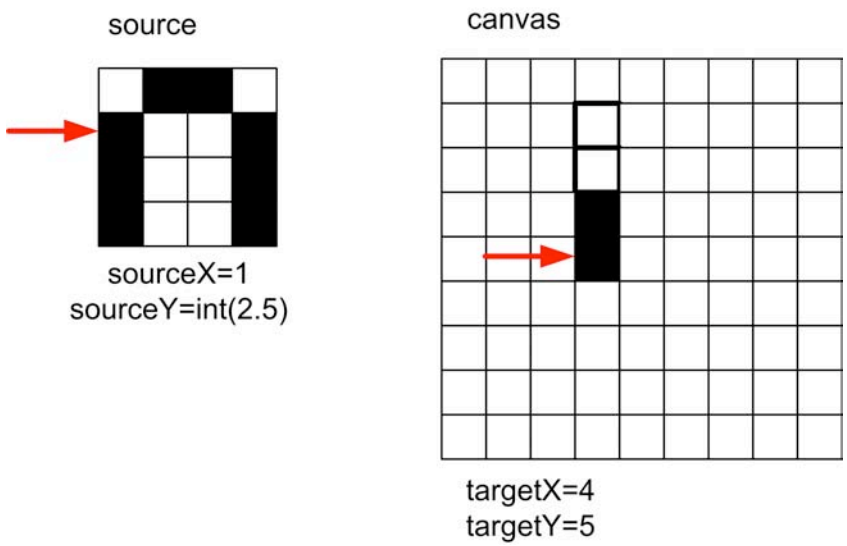
Scaling up: How it works 3

- Black pixels gets copied once…



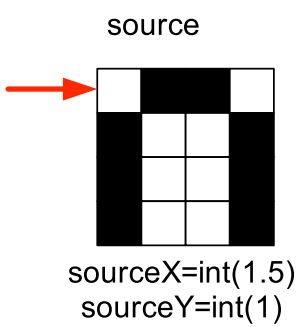
Scaling up: How it works 4

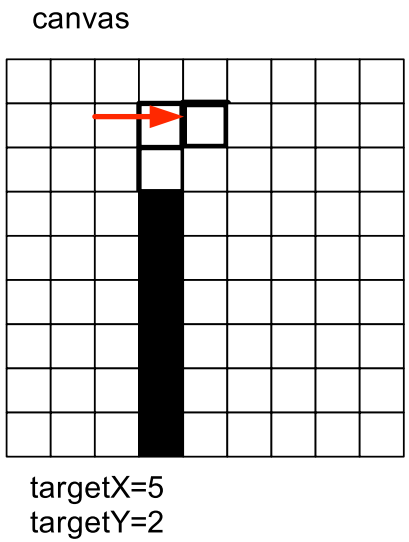
- And twice…



Scaling up: How it works 5

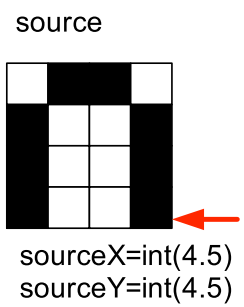
- The next “column” (x) in the source, is the *same* “column” (x) in the target.

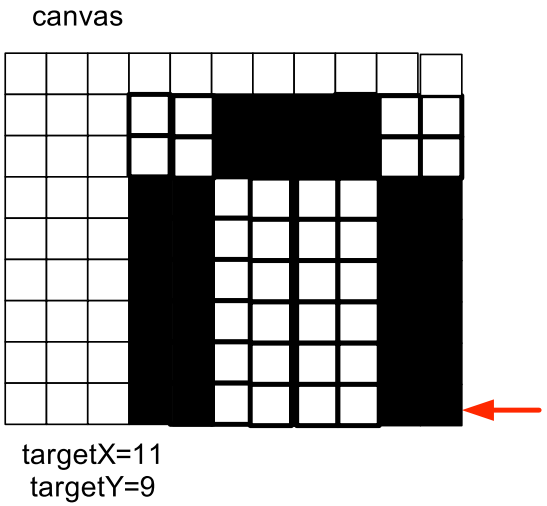




Scaling up: How it ends up

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





Coming Attractions

- For Monday
  - **Try to fix the scale-up example**
  - **Read Chapter 4.3-4.6**
  - **Do Quiz**
- Friday
  - **Assignment 3 Due**
  - **Group project 1.1 due**