CS 1124
Media Computation

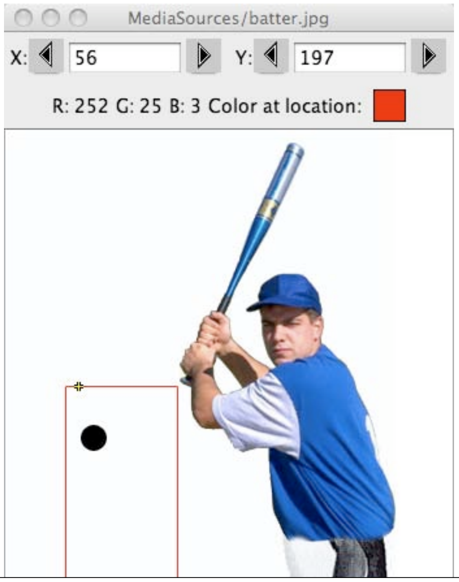
Steve Harrison
Lecture 5.1 (September 22, 2008)

Today

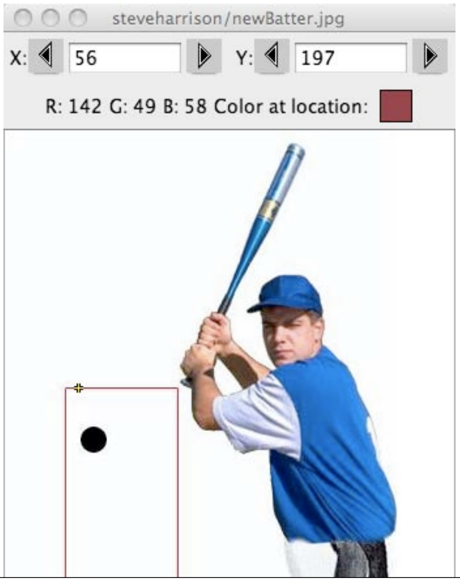
- Losses from JPEG compression
- Blending pictures together
  - **blend 1 mix two pictures together (DONE)**
  - **blend 2 (from the book) overlap two pictures (DID YOU DO IT ALREADY?)**
  - **blend 3 (iTunes) mirror effect**
- Scaling (again)
- Class/group project for Friday

When you write out a picture, read it back in, why are the RGB values changed?

```
>>> batterFile = pickAFile()
>>> batterPic = makePicture( batterFile )
>>> writePictureTo( batterPic, "newBatter.jpg" )
>>> newBatterPic = makePicture( pickAFile() )
```



R: 252 G: 25 B: 3 Color at location: 🟥



R: 142 G: 49 B: 58 Color at location: 🟥

Look at the red line of the strike zone.
And neither are (255,0,0) !

Why did it happen?

- JPEG
  - **low quality setting --> look OK, but is not same picture**

What can we do?

- Do “full quality” JPEG
- Change to a better format
  - .png
  -

Today

- Losses from JPEG compression
- Blending pictures together
    - **blend 1 mix two pictures together (DONE)**
    - **blend 2 (from the book) overlap two pictures (DID YOU DO IT ALREADY?)**
    - **blend 3 (iTunes) mirror effect**
- Scaling (again)
- Class/group project for Friday

Blending pictures together (1)

- 50% of picture + 50% of another = blended image!
  - **works on a pixel-by-pixel / color-by-color basis!**
- psuedo code
  - **a "program" made of comments**
  - **a template to write the program**

- blend 1 (file1, file2)
  - **get the pictures in each file**
  - **make a canvas for blended picture**
  - **for each pixel add 50% of each color picture1 to 50% of each color of picture2, put into canvas**

Blending pictures together (1)

```
def blendTwoPictures( fileName1, filename2 ):
    # get the pictures in each file
    source1 = makePicture( fileName1 )
    source2 = makePicture( fileName2 )
    # get the least width and height (Why?)
    canvasX = min( getWidth( source1), getWidth(  source2 ) ) + 1
    canvasY = min( getHeight( source1), getHeight(  source2 ) ) + 1
    # make a canvas for the blended file
    canvas = makeEmptyPicture( canvasX, canvasY )
    # for each pixel add 50% of each color picture1 to 50% of each color of picture2, put
    into canvas
    for x in range(1, canvasX ) :
        for y in range( 1, canvasY ) :
            source1Pixel = getPixel( source1, x, y )
            source2Pixel = getPixel( source2, x, y ) )
            blendRed = (getRed( source1Pixel) * 0.5) + (getRed(source2Pixel) * 0.5)
            blendGreen = (getGreen( source1Pixel) * 0.5) + (getGreen(source2Pixel) * 0.5)
            blendBlue = (getBlue( source1Pixel) * 0.5) + (getBlue(source2Pixel) * 0.5)
            blendColor = makeColor( blendRed, blendGreen , blendBlue )
            setColor( getPixel( canvas, x, y ), blendColor )
    return canvas
```

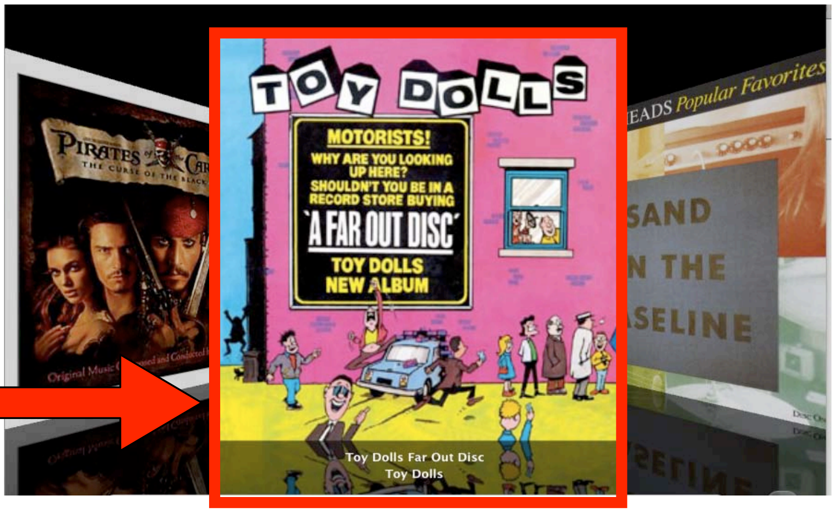
The shiny floor….

- iTunes album cover
- Do this
- Hierarchical decomposition?
- Psuedo code
  - iTunesEffect(fileName)

```
# get the picture, its height and create picture 50% taller picture
# copy the picture
# now put fading mirror image below picture
```

  - copyPicture(source, target, startX, startY)

```
# initialize target x and y to startX and startY
# for each pixel in the source, copy the pixel to the same location in the target
```



iTunesEffect( )

- Psuedo code (continued)
  - mirrorFade(source, target, startX, startY)

```
# set source y to last row so that we copy from bottom to top for mirror effect
# for each y in the target from the startY to the height of the target
    # figure out how much to fade to black for this row
    # for each x in the target from the startX to the width of the target
        # get the pixel from the source picture
        # multiply each color by the fade factor
        # put the pixel into the target
    # decrement the row in the source file to move towards the top of the source
```

- Notice that
  - **put x loop inside y loop to minimize # of calculations (Why?)**
  - **x is always the same for source and target !**

High level

```
def iTunesEffect(fileName):
   # get the picture, its height and create picture 50% taller picture
   source = makePicture( fileName )
   sourceHeight = getHeight( source )
   target = makeEmptyPicture( getWidth(source), int( sourceHeight*1.5 ) )
   # copy the picture
   target = copyPicture( source, target, 1, 1 )
   # now put fading mirror image below picture
   target = mirrorFade( source, target, 1, sourceHeight )
   show( target )
   return target
```

Lower level: copyPicture(s,t,x,y)

```
def copyPicture(src, trgt, startX, startY):
    # initialize target x and y to startX and startY
    # for each pixel in the source, copy the pixel to the same location in the target
    trgtX = startX
    for x in range(1, getWidth( src ) + 1 ) :
        trgtY = startY
        for y in range( 1, getHeight( src ) + 1 ) :
            setColor( getPixel( trgt, trgtX, trgtY ), getColor( getPixel( src, x, y ) ) )
            trgtY = trgtY + 1
        trgtX = trgtX + 1
    return trgt
```

Lower level: mirrorFade(s,t,x,y)

```
def mirrorFade(src, trgt, startX, startY):
    # set source y to last row so that we copy from bottom to top for mirror effect
    srcHeight = getHeight( src ) * 1.0
    srcY = srcHeight
    # for each y in the target from the startY to the height of the target
    for trgtY in range(startY, getHeight( trgt ) + 1 ) :
        # figure out how much to fade to black for this row
        fade = srcY  / srcHeight
        # for each x in the target and the source from the startX to the width of the pictures
        for x in range( startX, getWidth( src ) + 1 ) :
            # get the pixel from the source picture
            srcPixel = getPixel( src, x, int(srcY ) )
            # multiply each color by the fade factor
            trgtRed = int( getRed( srcPixel ) * fade)
            trgtGreen = int( getGreen( srcPixel ) * fade )
            trgtBlue = int( getBlue( srcPixel ) * fade )
            # put the pixel into the target
            setColor( getPixel( trgt, x, trgtY ), makeColor( trgtRed, trgtGreen, trgtBlue ) )
        # decrement the row in the source file to move towards the top of the source
        srcY = srcY - 1.0
    return trgt
```

Lower level: mirrorFade(s,t,x,y) alternatives

```
def mirrorFade(src, trgt, startX, startY):
    # set source y to last row so that we copy from bottom to top for mirror effect
    srcHeight = getHeight( src ) * 1.0
    srcY = srcHeight
    # for each y in the target from the startY to the height of the target
    for trgtY in range(startY, getHeight( trgt ) + 1 ) :
        # figure out how much to fade to black for this row
        fade = (srcY  / srcHeight) - 0.25 <== subtracting a factor
        # for each x in the target and the source from the startX to the width of the pictures
        for x in range( startX, getWidth( src ) + 1 ) :
            # get the pixel from the source picture
            srcPixel = getPixel( src, x, int(srcY ) )
            # multiply each color by the fade factor
            trgtRed = int( getRed( srcPixel ) * fade)
            trgtGreen = int( getGreen( srcPixel ) * fade )
            trgtBlue = int( getBlue( srcPixel ) * fade )
            # put the pixel into the target
            setColor( getPixel( trgt, x, trgtY ), makeColor( trgtRed, trgtGreen, trgtBlue ) )
        # decrement the row in the source file to move towards the top of the source
        srcY = srcY - 2.0 <== stepping by twos makes floor seem more oblique to viewer
        if srcY < 1.0 :
            srcY = 1.0
    return trgt
```

Today

- Blending pictures together
  - **blend 1 mix two pictures together (DONE)**
  - **blend 2 (from the book) overlap two pictures (DID YOU DO IT ALREADY?)**
  - **blend 3 (iTunes) mirror effect**
- Scaling (again)
- Doin' the directory thing in project 3
- Class/group project for Friday

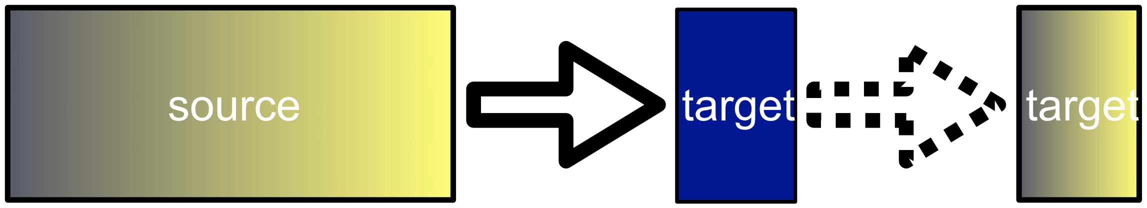
Scaling (again)

- Why should we figure out how to scale?
- Can calculate source(x,y) from target(x,y)

```
def scale( source, target ) :
  srcWid = getWidth( source )
  srcHit = getHeight( source )
  trgtWid = getWidth( target ) * 1.0
  trgtHit = getHeight( target ) * 1.0
  for x in range( 1, int( trgtWid + 1 ) ) :
    sourceX = int( (x / trgtWid * srcWid ) + .5 )
    if sourceX < 1 :
      sourceX = 1
    for y in range( 1, int( trgtHit + 1 ) ) :
     sourceY = int( (y / trgtHit * srcHit ) + .5 )
     if sourceY < 1 :
      sourceY = 1
     setColor( getPixel( target, x, y ), getColor( getPixel( source, sourceX, sourceY ) ) )
  return target
```

Scaling (again)

- How this works:



```
def scale( source, target ) :
  srcWid = getWidth( source )
  srcHit = getHeight( source )
  trgtWid = getWidth( target ) * 1.0
  trgtHit = getHeight( target ) * 1.0
  for x in range( 1, int( trgtWid + 1 ) ) :
    sourceX = int( (x / trgtWid * srcWid ) + .5 )
    if sourceX < 1 :
      sourceX = 1
    for y in range( 1, int( trgtHit + 1 ) ) :
     sourceY = int( (y / trgtHit * srcHit ) + .5 )
     if sourceY < 1 :
      sourceY = 1
     setColor( getPixel( target, x, y ), getColor( getPixel( source, sourceX, sourceY ) ) )
  return target
```

Today

- Losses from JPEG compression
- Blending pictures together
  - **blend 1 mix two pictures together (DONE)**
  - **blend 2 (from the book) overlap two pictures (DID YOU DO IT ALREADY?)**
  - **blend 3 (iTunes) mirror effect**
- Scaling (again)
- Class/group project for Friday

Grading the Group Project (visual)

- By 2:00 PM Friday
  - e-mail me <srh@vt.edu> code, pictures, and names of people in your group
- Bring to Lab for demo to class
  - if reasonable, we'll try using your abstraction with the results of other groups.
- Everyone in group gets same grade
  - unless you tell me otherwise
- Rubric: creativity of idea: 10%, results: 30%, teamwork: 30%, modularity: 20%, difficulty: 10%

Coming Attractions

- This Friday (9/26)
  - Group project due 2:00 PM
  - Bring to Lab!
- Wednesday (9/24)
  - Play with iTunes effect / bring better fading results
  - midterm practice quiz opens -- NOT GRADED
- Next Monday (9/29)
  - Assignment 4 due 10:00 AM
- Next Wednesday (10/1)
  - midterm