Conditionally Modifying Pixels

Barb Ericson
Oct 2010

Learning Goals

- Understand at a conceptual and practical level
  - How to conditionally execute a statement or a block of statements using *if*
  - How to use a conditional with two possible results: *if* and *else*
  - How to use a conditional with more than two possible results: *if*, *else if*, and *else*
  - How to use the not (!) operator
  - How to combine Boolean expressions with *and* and *or*

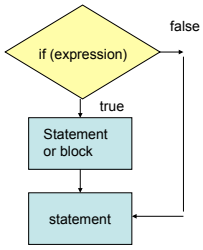
Conditionally Modifying Pixels

- Up to now we have modified all the pixels in a picture or in a range the same way
- But, sometimes we want to modify only pixels that meet a certain condition
  - Like when a person has "red eye"
  - We only want to modify the pixels in an area that have a value close to red

Conditional Execution

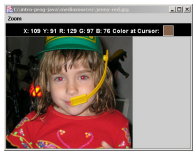
- Sometimes we want a statement or block of statements executed only if some expression is true
  - We can use the "if" statement in Java

```
if (colorDistance < value)
    Statement or block to
      execute
next statement
```



Remove Red Eye

- Red eye is when the flash from the camera is reflected from the subject's eyes
- We want to change the red color in the eyes to another color
  - But not change the red of her dress



Red Eye Algorithm

- We can find the area around the eyes to limit where we change the colors
  - Using pictureObj.explore()
  - But we still just want to change the pixels that are "close to" red.
  - We can find the distance between the current color and our definition of red
    - And change the color of the current pixel only if the current color is within some distance to the desired color

Detailed Red Eye Algorithm

- Loop with x staring at some passed start value and while it is less than some passed end value
  - Loop with y starting at some passed start value and while it is less than some passed end value
    - Get the pixel at this x and y
    - Get the distance between the pixel color and red
    - If the distance is less than some value (167) change the color to some passed new color

Blocks of Statements

- The if statement will conditionally execute
  - the following statement or
  - A block of statements
  - if the Boolean expression is true
- To conditionally execute a block of statements
  - Enclose them in '{' and '}'
- Indent the following statement or block of statements
  - To make it easier to read
- It is good practice to always enclose conditional statements in a block
  - Less likely to cause an error if the code is modified

Color Distance

- The distance between two points is computed as
  - Square root of $((x1 – x2)^2 + (y1 – y2)^2)$
- The distance between two colors can be computed
  - Square root of $((red1 – red2)^2 + (green1-green2)^2 + (blue1 – blue2)^2)$
  - There is a method in the Pixel class to do this
    - double dist = pixelObj.colorDistance(color1);

Remove red eye method

```
public void removeRedEye(int startX, int startY, int endX,
                         int endY, Color newColor)
 {
   Pixel pixel = null;

   /* loop through the pixels in the rectangle defined by the
      startX, startY, and endX and endY */
   for (int x = startX; x < endX; x++)
   {
     for (int y = startY; y < endY; y++)
     {
       // get the current pixel
       pixel = getPixel(x,y);
```


Remove Red Eye Method

```
      // if the color is near red then change it
      if (pixel.colorDistance(Color.red) < 167)
        pixel.setColor(newColor);
    }
   }
 }
```


Testing removeRedEye

- Try the following to test removeRedEye

```
> String fileName =
  "c:/intro-prog-java/mediasources/jenny-red.jpg";
> Picture jennyPicture = new Picture(fileName);
> jennyPicture.removeRedEye(109,91,202,107,
                            java.awt.Color.black);
> jennyPicture.explore();
```

Challenge

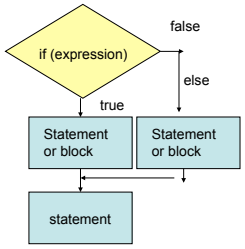
- Take a picture of a friend or find a picture on the web
  - And try to change their eye color
  - Try to change their hair color
  - Try to change their clothing color
- Can you write one method to do this?
  - And call it several times with different parameters?

A Conditional with 2 Outcomes

- You might want to do one thing if a Boolean expression is true and another if it is false
  - int x = 30;
  - if (x < 30) System.out.println("x < 30");
  - if (x >= 30) System.out.println("x >= 30");
- But, then you have to test 2 Boolean expressions
  - Instead you can use an if with an else

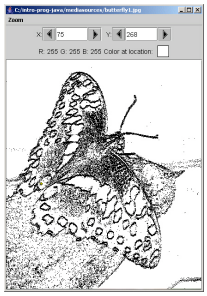
Use if and else for two possibilities

```
int x = 30
if (x < 30)
  System.out.println("x<30");
else
   System.out.println("x>=30");
```



Edge Detection

- Loop through all the pixels in the picture, except the last row
  - Calculate the average color for the current pixel and the pixel at the same x but y+1.
  - Get the distance between the two averages
  - If the absolute value of the distance is greater than some value turn the current pixel black
  - Otherwise turn the current pixel white



Edge Detection Algorithm

- To find areas of high contrast
  - Try to loop from y = 0 to y = height – 1
    - Loop from x = 0 to x = width
      - Get the pixel at the x and y (top pixel)
      - Get the pixel at the x and (y + 1) bottom pixel
      - Get the average of the top pixel color values
      - Get the average of the bottom pixel color values
      - If the absolute value of the difference between the averages is over a passed limit
        - » Turn the pixel black
        - » Otherwise (else) turn the pixel white

Edge detection method

```
public void edgeDetection(double amount) {
   Pixel topPixel = null;
   Pixel bottomPixel = null;
   double topAverage = 0.0;
   double bottomAverage = 0.0;
   int endY = this.getHeight() - 1;

   /* loop through y values from 0 to height - 2
    * (since compare to below pixel) */
   for (int y = 0; y < endY; y++) {
    // loop through the x values from 0 to width
    for (int x = 0; x < this.getWidth(); x++) {
```

Edge detection method - cont

```
// get the top and bottom pixels
topPixel = this.getPixel(x,y);
bottomPixel = this.getPixel(x,y+1);

// get the color averages for the two pixels
topAverage = topPixel.getAverage();
bottomAverage = bottomPixel.getAverage();

/* check if the absolute value of the difference
 * is less than the amount */
if (Math.abs(topAverage - bottomAverage) < amount) {
  topPixel.setColor(Color.WHITE);
```


Edge detection method - cont

```
      // else set the color to black
      } else {
        topPixel.setColor(Color.BLACK);
      }
    }
  }
}
```

Test with:

```
> String fileName = FileChooser.getMediaPath("butterfly1.jpg");
> Picture p = new Picture(fileName);
> p.explore();
> p.edgeDetection(10);
> p.explore();
```


Where to put the curly braces?

- You may have noticed that in edgeDetection the opening curly brace '{' is at the end of the statement

```
for (int y=0; y < endY; y++) {
```

- rather than on a new line

```
for (int y=0; y < endY; y++)
{
```

- It doesn't matter to Java
  - The Java standard says put them at the end of the line

Challenge

- Create another method for simple edge detection
  - This time compare the current pixel with the one to the right (x+1)
  - How do you need to change the nested loop?
  - Do you get a different result?

Testing for not true

- We usually use a conditional to do some action when a Boolean expression is true
  - But, what about if we want to do some action when a Boolean expression is false?
  - If the contrast is not low then set it to black else set it to white
- We can use an if with the Boolean expression negated using '!'

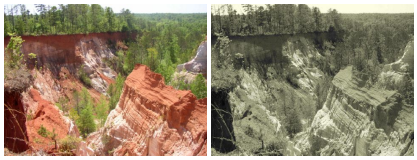
```
> !true
false
```


Challenge

- You can use ! to check if something is not true
  - !(20 < x) will be true when 20 is greater than or equal to x
- Modify the edgeDetection method to use !
  - Set the pixel to black when the contrast is not low, else to white

Sepia-Toned Pictures

- Have a yellowish tint, used to make things look old and western

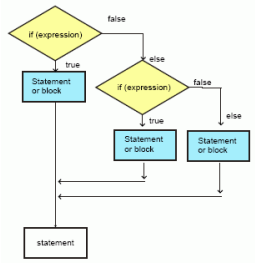


Sepia-toned Algorithm

- First make the picture grayscale.
  - Calling the method you created
- Loop through all the pixels in the picture
  - Change the shadows (darkest grays) to be even darker (0 <= red < 60)
  - Make the middle grays a brown color (60 <= red < 190)
  - Make the highlights (lightest grays) a bit yellow (red >= 190)
    - Increase red and green
    - Or decrease blue

Conditionals with > 2 Choices

```
int y = 2;
if (y < 10)
  System.out.println(
            "y is less than 10");
else if (y == 10)
  System.out.println(
            "y is equal to 10");
else
  System.out.println(
            "y is greater than 10");
```



Sepia tint method

```
public void sepiaTint()
 {
   Pixel pixel = null;
   double redValue = 0;
   double greenValue = 0;
   double blueValue = 0;

   // first change the current picture to grayscale
   this.grayscale();

   // loop through the pixels
   for (int x = 0; x < this.getWidth(); x++)
   {
```


Sepia tint method - cont

```
    for (int y = 0; y < this.getHeight(); y++)
    {
      // get the current pixel and color values
      pixel = this.getPixel(x,y);
      redValue = pixel.getRed();
      greenValue = pixel.getGreen();
      blueValue = pixel.getBlue();

      // tint the shadows darker
      if (redValue < 60)
      {
        redValue = redValue * 0.9;
        greenValue = greenValue * 0.9;
        blueValue = blueValue * 0.9;
      }
```


Sepia tint method - cont

```
      // tint the midtones a light brown
      // by reducing the blue
      else if (redValue < 190)
      {
        blueValue = blueValue * 0.8;
      }
      // tint the highlights a light yellow
      // by reducing the blue
      else
      {
        blueValue = blueValue * 0.9;
      }
```

Sepia tint method - cont

```
        // set the colors
      pixel.setRed((int) redValue);
      pixel.setGreen((int) greenValue);
      pixel.setBlue((int) blueValue);
    }
  }
}
```


Testing sepiaTint

```
> Picture picture =
      new Picture(Picture.getMediaPath("gorge.jpg"));
> picture.show();
> picture.sepiaTint();
> picture.repaint();
```


Posterize

- Reducing the number of different colors in an image
- Set all values in a range to one value (the midpoint of the range)
  - Set all < 64 to 31
  - Set all <128 to 95
  - Set all < 192 to 159
  - Set the rest to 223

Posterize Algorithm

- Loop through all the pixels in an image
  - Get the red value for the pixel
    - Find the right range and set the new red value
  - Get the green value for the pixel
    - Find the right range and set the new green value
  - Get the blue value for the pixel
    - Find the right range and set the new blue value

Posterize method

```
public void posterize()
 {
   Pixel pixel = null;
   int redValue = 0;
   int greenValue = 0;
   int blueValue = 0;

   // loop through the pixels
   for (int x = 0; x < this.getWidth(); x++) {
     for (int y = 0; y < this.getHeight(); y++) {

       // get the current pixel and colors
       pixel = this.getPixel(x,y);
```


Posterize method - cont

```
      redValue = pixel.getRed();
      greenValue = pixel.getGreen();
      blueValue = pixel.getBlue();

      // check for red range and change color
      if (redValue < 64)
        redValue = 31;
      else if (redValue < 128)
        redValue = 95;
      else if (redValue < 192)
        redValue = 159;
      else
        redValue = 223;
```

Posterize method - cont

```
// check for green range
if (greenValue < 64)
  greenValue = 31;
else if (greenValue < 128)
  greenValue = 95;
else if (greenValue < 192)
  greenValue = 159;
else
  greenValue = 223;
```


Posterize method - cont

```
// check for blue range
if (blueValue < 64)
  blueValue = 31;
else if (blueValue < 128)
  blueValue = 95;
else if (blueValue < 192)
  blueValue = 159;
else
  blueValue = 223;

// set the colors
pixel.setRed(redValue);
pixel.setGreen(greenValue);
pixel.setBlue(blueValue);
  }
 }
}
```

Test with tammy.jpg

Is there a better way?

- What posterize does is set up 4 levels
  - and set the pixel color to the midpoint of that level
- We can make a more general method
  - Check if the value is in a range
  - Set the value to the midpoint of that range
- How do you check if a value is in a range?
  - (bottomValue <= testValue < topValue)
  - if (bottomValue <= testValue && testValue < topValue)

Logical And and Or

- You can join two Boolean expressions with
  - And
    - Then both expressions must be true for the combined expression to be true
    - Use && for a logical And in Java
  - Or
    - Then just one of the expressions must be true for the combined expression to be true
    - Use || for a logical Or in Java

How many when there is an "Or"

- You need to help clean the house
  - You can clean the bathroom or the kitchen or the living room
  - How many jobs do you have to do?
- You want to get an ice cream
  - The flavors you can pick from are chocolate, vanilla, strawberry, or orange sherbet
  - How many flavors do you need to pick for a single scoop?

Truth Table

Conditional	Operand 1	Operand 2	Result
And	true	true	true
And	true	false	false
And	false	true	false
And	false	false	false
Or	true	true	true
Or	true	false	true
Or	false	true	true
Or	false	false	false
Exclusive Or	true	true	false
Exclusive Or	true	false	true
Exclusive Or	false	true	true
Exclusive Or	false	false	false

Conditional Operators

- We can check if several things are true - And
  - Using && (evaluation stops if the first item is false)
    - Called short circuit evaluation
  - Using & (to always evaluate both operands)
- We can check if at least one of several things are true - Or
  - Using || (evaluation stops if the first item is true)
  - Using | (to always evaluate both operands)
- We can check if only one and only one of the things is true – Exclusive Or
  - Using ^

Better Posterize Method

- In our current `posterize` method we checked for a range and set the color to the middle of that range
  - **if** (redValue < 60)
- But what if we want more or less ranges?
  - We would have to add or remove some of the conditionals
- What if we calculate the endpoints of the ranges?
  - If the value is between a bottomValue and topValue set it to the middleValue

Better Posterize Method

```
public void posterize(int numLevels)
 {
   Pixel pixel = null;
   int redValue = 0;
   int greenValue = 0;
   int blueValue = 0;
   int increment = (int) (256.0 / numLevels);
   int bottomValue, topValue, middleValue = 0;
   // loop through the pixels
   for (int x = 0; x < this.getWidth(); x++) {
     for (int y = 0; y < this.getHeight(); y++) {
       // get the current pixel and colors
       pixel = this.getPixel(x,y);
       redValue = pixel.getRed();
       greenValue = pixel.getGreen();
       blueValue = pixel.getBlue();
```


Better Posterize Method - cont

```
      // loop through the number of levels
      for (int i = 0; i < numLevels; i++)
      {
        // compute the bottom, top, and middle values
        bottomValue = i * increment;
        topValue = (i + 1) * increment;
        middleValue = (int) ((bottomValue + topValue - 1) / 2.0);

        /* check if current values are in current range and
         * if so set them to the middle value
         */
        if (bottomValue <= redValue &&
            redValue < topValue)
          pixel.setRed(middleValue);
        if (bottomValue <= greenValue &&
            greenValue < topValue)
          pixel.setGreen(middleValue);
        if (bottomValue <= blueValue &&
            blueValue < topValue)
          pixel.setBlue(middleValue);
      }
    }
  }
}
```


Challenge

- What are the values for bottomValue, topValue and middleValue when the numLevels is 2? What are the values when the numLevels is 4?
- Try the new `posterize` method out on a picture with 2 levels and with 4 levels.
  - What happens to the picture when you increase the number of levels?

Overloading

- We now have two `posterize` methods
  - One that takes no parameters
  - And one that takes the number of levels as an integer value
- It is okay to have more than one method with the same name
  - As long as the parameter list is different
    - Number of parameters
    - Types of parameters
    - Order of the types

Highlight Extremes Challenge

- Radiologists often miss things in x-rays that are too bright or too dark.
- Let's highlight all pixels in a picture that are close to black or white
  - Using a logical 'or' (||)
  - Using pixelObj.colorDistance

Using and to stop going out of bounds

- Did you get an java.lang.ArrayIndexOutOfBoundsException?
  - This means you tried to access an element of an array that was past the bounds of the array: like past the width or height of the picture
- You probably tried to work with two different size pictures
  - But only checked that the sourceX or targetX was in the range of one of the pictures
  - Modify the code to use && to check that the value is within the range of both pictures
    - sourceX < this.getWidth() && sourceX < pict2.getWidth()
    - sourceY < this.getHeight() && sourceY < pict2.getHeight()

Example Out of Bounds Error

```
> String fileName = FileChooser.getMediaPath("caterpillar.jpg");
> Picture p = new Picture(fileName);
> System.out.println(p.getWidth());
329
> System.out.println(p.getHeight());
150
> p.getPixel(330,160); // 330 is past the width of the picture and 160 is past the height of the picture
java.lang.ArrayIndexOutOfBoundsException: Coordinate out of
bounds!
  at sun.awt.image.ByteInterleavedRaster.getDataElements
    (Unknown Source)
  at java.awt.image.BufferedImage.getRGB(Unknown Source)
  at SimplePicture.getBasicPixel(SimplePicture.java:247)
  at Pixel.setValuesFromPictureAndLocation(Pixel.java:137)
  at Pixel.<init>(Pixel.java:57)
  at SimplePicture.getPixel(SimplePicture.java:270)
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
  at sun.reflect.DelegatingMethodAccessorImpl.invoke
    (Unknown Source)
  at java.lang.reflect.Method.invoke(Unknown Source)
```


Challenge

- Modify the general copy method in the last chapter to work even if where you are trying to copy the source picture past the width or height of the target picture
  - Fix the for loop to stop when you reach the width or height of either picture

Blurring a Picture

- When we scale a picture up the picture can become pixelated
  - Jagged edges instead of smooth edges
  - We can smooth out the picture by setting a pixel's color values to the average of the surrounding pixels
    - All pixels surrounding the current pixel
  - We need to be careful that we don't end up outside the array
    - Guard using x >= 0 && x < this.getWidth && y >= 0 && y < this.getHeight()

Blur Method

```
public void blur(int numPixels)
 {
   Pixel pixel = null;
   Pixel samplePixel = null;
   int redValue = 0;
   int greenValue = 0;
   int blueValue = 0;
   int count = 0;

   // loop through the pixels
   for (int x=0; x < this.getWidth(); x++)
   {
    for (int y=0; y < this.getHeight(); y++)
    {
     // get the current pixel
     pixel = this.getPixel(x,y);
     // reset the count and red, green, and blue values
     count = 0;
     redValue = greenValue = blueValue = 0;
```

Blur Method - cont

```
/* loop through pixel numPixels before x to
 * numPixels after x
 */
for (int xSample = x - numPixels;
     xSample <= x + numPixels;
     xSample++) {
  for (int ySample = y - numPixels;
       ySample <= y + numPixels;
       ySample++) {
    /* check that we are in the range of acceptable
     * pixels
     */
    if (xSample >= 0 && xSample < this.getWidth() &&
        ySample >= 0 && ySample < this.getHeight()) {
      samplePixel = this.getPixel(xSample,ySample);
      redValue = redValue + samplePixel.getRed();
      greenValue = greenValue + samplePixel.getGreen();
      blueValue = blueValue + samplePixel.getBlue();
      count = count + 1;
```


Blur Method - cont

```
        }
      }
    }
    // use average color of surrounding pixels
    Color newColor = new Color(redValue / count,
                               greenValue / count,
                               blueValue / count);
    pixel.setColor(newColor);
  }
}
```


Testing the Blur Method

```
> Picture p = new Picture(
    FileChooser.getMediaPath("flower1.jpg"));
> p = p.scaleUp(2);
> p.explore();
> p.blur(2);
> p.explore();
```


Challenge

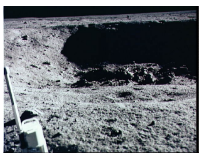
- The blur method modifies the pixels that are used in later calculations
  - Affecting the result
- Instead create a new picture the same width and height of the original picture and set the color in that picture based on the colors in the original picture
  - Remember to return the resulting picture
  - Remember to save a reference to the resulting picture

Background Replacement

- If you have a picture of a person in front of some background
- And a picture of the background itself
- Can you replace the pixel colors for the background to be from another image?

Replace Background

- Replace the colors at all the pixels in the source image
  - That are within some range of the background color
  - Use pixels from another background image



Replace Background Algorithm

- Works on the source picture
  - Pass in the original background and the new background pictures
- Loop through all the pixels in the source image
  - Check if the distance from the source pixel color is within 15.0 of the background picture pixel color
    - If so replace it with the color at the new background pixel

Replace Background Method

```
public void swapBackground(Picture oldBackground,
                           Picture newBackground)
 {
   Pixel currPixel = null;
   Pixel oldPixel = null;
   Pixel newPixel = null;

   // loop through the columns
   for (int x = 0; x < getWidth(); x++)
   {
     // loop through the rows
     for (int y = 0; y < getHeight(); y++)
     {
```


Swap Background - Cont

```
      // get the current pixel and old background pixel
      currPixel = this.getPixel(x,y);
      oldPixel = oldBackground.getPixel(x,y);

      /* if the distance between the current pixel color
       * and the old background pixel color is less
       * than the 15 then swap in the new background pixel
       */
      if (currPixel.colorDistance(oldPixel.getColor()) < 15.0)
      {
        newPixel = newBackground.getPixel(x,y);
        currPixel.setColor(newPixel.getColor());
      }
    }
  }
}
```


Testing swapBackground

```
> String fileName =
        FileChooser.getMediaPath("kid-in-frame.jpg");
> Picture p = new Picture(fileName);
> fileName = FileChooser.getMediaPath("bgframe.jpg");
> Picture oldBg = new Picture(fileName);
> fileName = FileChooser.getMediaPath("moon
  -surface.jpg");
> Picture newBg = new Picture(fileName);
> p.swapBackground(oldBg,newBg);
> p.show();
```


Replace Background Result

- The background color was too close to the shirt color


Chromakey – Blue Screen

- For TV and movie special effects they use a blue or green screen
  - Here just a blue sheet was used
  - Professionally you need an evenly lit, bright, pure blue background
    - With nothing blue in the scene

Chromakey Algorithm

- Loop through all the pixels in the current picture
  - If the current pixel color is close to blue
  - Replace that pixel color with the pixel at the same location on the new background picture
    - the new background picture must be at least as big as the current picture

Chromakey method

```
public void chromakey(Picture newBg)
 {
   Pixel currPixel = null;
   Pixel newPixel = null;

   // loop through the columns
   for (int x = 0; x < getWidth(); x++)
   {
    // loop through the rows
    for (int y = 0; y < getHeight(); y++)
    {
      // get the current pixel
      currPixel = this.getPixel(x,y);
```


Chromakey method - cont

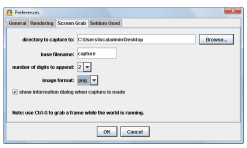
```
      /* if the color at the current pixel is mostly blue
       * (blue value is greater than red and green
       * combined), then use new background color
       */
      if (currPixel.getRed() + currPixel.getGreen() <
          currPixel.getBlue())
      {
          newPixel = newBg.getPixel(x,y);
          currPixel.setColor(newPixel.getColor());
      }
    }
  }
}
```


Testing chromakey

```
> String fileName = FileChooser.getMediaPath("blue-mark.jpg");
> Picture mark = new Picture(fileName);
> fileName = FileChooser.getMediaPath("moon-surface.jpg");
> Picture newBg = new Picture(fileName);
> mark.chromakey(newBg);
> mark.explore();
> mark = new Picture(
  FileChooser.getMediaPath("blue-mark.jpg"));
> newBg = new Picture(FileChooser.getMediaPath("beach.jpg"));
> mark.chromakey(newBg);
> mark.explore();
```

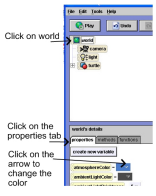

Put an Alice Character on a Picture

- Position an Alice character in a world in Alice
- Change the screen grab to use png format instead of jpg
  - png is not a lossy compression format and jpg is
  - Edit – Preferences – Screen Grab – image format



Setting up the Picture (cont)

- Delete the ground in the Alice world
  - Right click on it and select delete
- Click on World in the Object tree and then click on the Property tab
  - Change the atmosphere color to green (or blue if the Alice character has green in it)



Save a Picture from Alice

- Click the Play button
- Drag to make the window smaller or you can also set the window size using
  - Edit – Preferences – Rendering – Width or Height
- Click the "Take Picture" button
  - This writes a picture to the desktop: capture00.png then capture01.png, etc
  - Rename the picture and put it in mediasources
  - Copy just the non-green pixels to another picture

Challenge

- Create a method that copies just the pixels in a picture that aren't close to a particular color to another picture
  - Use this to copy an Alice character to a normal picture
  - You can specify where to copy it to on the resulting picture
  - You can specify the upper left corner in the source to start the copy from
  - You can specify the bottom right corner in the source to stop at

Summary

- You can conditionally execute code
  - using an **if**
- You can have two outcomes
  - using **if** and **else**
- You can have more than two outcomes
  - using **if**, **else if**, and **else**
- You can test for a condition being false
  - using **!**
- You can combine Boolean expressions
  - using **&&** for a logical "And" and **||** for "Or"