colorPointer

Superficially our second Processing program, `colorPointer`, is very similar to `firstPointer`. The only change is the `mouseClicked` routine, which now prints information about the color of the pixel at which the user clicked. The new routine is shown below. You should run and try this program in Processing to see what it does.

```
void mouseClicked()              // The mouseClicked routine is run each time the user clicks the mouse
{                                // This routine is only needed to respond to mouse clicks.
  color pixel;                   // Used for pixel information.
  print("You clicked at x = "); // These lines print to the console below.
  print(mouseX);                 // mouseX and mouseY are the mouse coordinates.
  print(" y = ");                // The print function prints to the console without a carriage return.
  print(mouseY);                 // The println function prints to the console with a carriage return.
  pixel = photo.pixels[mouseY * width + mouseX];   // Get pixel information at point.
  print("  red = ");             // Print the RGB values for this pixel.
  print(red(pixel));
  print("   green = ");
  print(green(pixel));
  print("   blue = ");
  println(blue(pixel));
}
```

Even though, superficially, this program looks very similar to our first program, `firstPointer`, it actually introduces three new ideas:

- `firstPointer` used only global variables, which are shared by one or more routines. Complex programs with many different routines use many different variables, some of which are shared or "global" variables and others of which are "local" variables that are used only by one routine. This new `mouseClicked` routine uses the local variable `pixel`. This variable is of type `color`.
- The global variable `photo` of type `PImage` contains color information about each of the pixels in the image. This information is in the form of an array with one element of type color for each pixel. The array `photo.pixels` is this array. The value of the color variable for the pixel whose coordinates are (x, y) is the $(y * width + x)^{th}$ element of this array. The line

  ```
  pixel = photo.pixels[mouseY * width + mouseX];   // Get pixel color at point.
  ```

  puts this information for the pixel at which the user clicked in the variable `pixel`, which we recall is of type `color`.

- The three most immediate qualities of a color are its red, green, and blue components. You can think[1] of each pixel in the display area as having three tiny LEDs, a red LED, a green LED, and a blue LED. Each LED shines at a different intensity. The number 0 represents the lowest level, off, of intensity and the number 255 represents the brightest possible intensity. The color you see is produced by mixtures of red, green and blue. The code fragments `red(pixel)`, `green(pixel)` and `blue(pixel)` extract these three intensities.

Play with the program `colorPointer` to see the RGB (red, green, blue) components of the various colors in the image.

[1] This is a useful simplifcation. In the same way you can think of each pixel on the sensor of your digital camera as having being made up of a red sensor, a green sensor, and a blue sensor. Both simplifcations are useful but wrong.