Images Recap

- declare a variable to hold an image

  `PImage` *`img`*`;`

- load an image (image must also be added to sketch with Sketch → Add File...)

  `img`` = loadImage(“`*`myimage.jpg`*`”);`

- displaying an image

  ```
  image(img,x,y);
  image(img,x,y,w,h);
  ```

  (use `imageMode(CORNER)` or `imageMode(CENTER)` to specify whether (x,y) is the upper left corner or center of the image)

- size of an image

  ```
  img.width
  img.height
  ```

- loading pixels

  `img``.loadPixels();`

- accessing the color of a pixel (once loaded)

  `img``.pixel[`*`loc`*`]`

  (loc = row**img*.width+col)

- accessing color components for a pixel (once loaded)

  ```
  red(img.pixel[loc])
  green(img.pixel[loc])
  blue(img.pixel[loc])
  ```

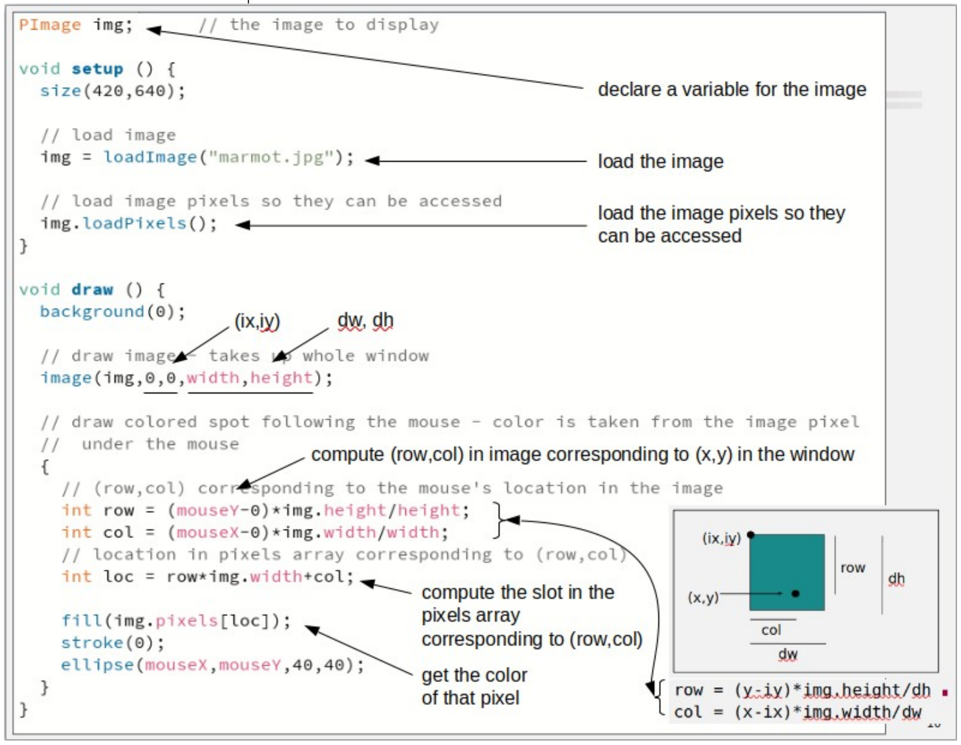
  (loc = row**img*.width+col)

From (x,y) To (row,col)

- (x,y) is the coordinate in the drawing window
- (row,col) is the pixel coordinate in the image
- the area covered by the image is dw × dh and has upper left corner (ix,iy)
- the image's dimensions are *img*.width × *img*.height

`row =(y-iy)*`*`img`*`.height/dh`

`col = (x-ix)*`*`img`*`.width/dw`



```
PImage img;        // the image to display

void setup () {
  size(420,640);

  // load image
  img = loadImage("marmot.jpg");

  // load image pixels so they can be accessed
  img.loadPixels();
}

void draw () {
  background(0);

  // draw image - takes up whole window
  image(img,0,0,width,height);

  // draw colored spot following the mouse - color is taken from the image pixel
  //  under the mouse
  {
    // (row,col) corresponding to the mouse's location in the image
    int row = (mouseY-0)*img.height/height;
    int col = (mouseX-0)*img.width/width;
    // location in pixels array corresponding to (row,col)
    int loc = row*img.width+col;

    fill(img.pixels[loc]);
    stroke(0);
    ellipse(mouseX,mouseY,40,40);
  }
}
```

```
row = (y-iy)*img.height/dh
col = (x-ix)*img.width/dw
```

declare a variable for the image
load the image
load the image pixels so they can be accessed
(ix,iy)
dw, dh
compute (row,col) in image corresponding to (x,y) in the window
compute the slot in the pixels array corresponding to (row,col)
get the color of that pixel
(ix,iy)
(x,y)
row
dh
col
dw

At the End of Class

Hand in whatever you have done during class, even if a sketch is incomplete.

- Make sure each sketch is named as directed and has a comment with the names of your group. Also be sure to save your sketches! (in Linux, this should be in your sketchbook **~/cs120/sketchbook**)
- Copy the entire directory for each sketch (not only the .pde file) into your handin directory (**/classes/cs120/handin/*username***). You only need to hand in one copy for the group. (If you are running Processing on your computer instead of using the Linux virtual desktop, you will need to use FileZilla to copy the sketches.)

Exercises

For all sketches, be sure to **include a comment with the names of your group at the beginning of the sketch.**

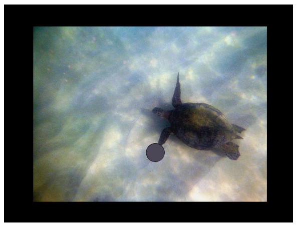
Provided images –

- If you are working on your own computer, use the link on the schedule page – right-click on the filename and choose "Save Link As…" to save the images you want to use on your computer.
- If you are working in Linux (either the virtual desktop or on an actual machine), you can find the same images in **/classes/cs120/images**.

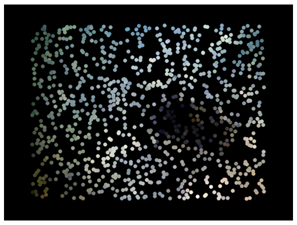
Don't forget to also add the image(s) to each sketch you create with Sketch → Add File… so that your sketch can find the images when it runs.

---

1. Create a new sketch called **sketch_241120a** and paste in the contents of the `starter_dot` sketch from the schedule page.   You can keep using the same sea turtle image or switch to something else – either way, don't forget to add the image you are using to your sketch with Sketch → Add File…
2. Modify the sketch so that the image is displayed with a size smaller than the drawing window – there should be a black border visible on all sides of the image (this is just the background showing, you don't need to draw something additional) and the colored dot must only be displayed when the mouse is over the image.



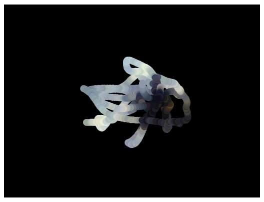
3. Save a copy of your sketch as **sketch_241120b**, then modify the copy so the color of the dot uses only the red component of the image pixel's color. (The green and blue components for the dot's color should be 0.)
4. Create a new sketch called **sketch_241120c** where the picture is filled in with an animation of dots. (One dot should be drawn in each frame.) This is not much different from #2 – start by pasting in a copy of that solution, then remove the drawing of the image, choose a random position for the dot instead of using the mouse position (only choose positions overlapping the image), make the dots smaller, and don't clear the background in every frame.



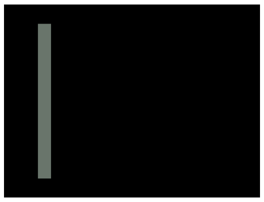
5. Save a copy of the previous sketch as **sketch_241120d**, then modify the copy to draw a large number of random dots (multiple thousands) in each frame. Use `randomSeed()` in `draw()` to avoid flickering, though the flickering is also a neat effect.

If you have time –

- Create a sketch **sketch_241120e** where a dot whose position is determined by Perlin noise and whose color comes from the image wanders around the drawing window. (This will be very similar to #4, but with the dot position coming from Perlin noise instead of being random. And you might want the the dot to be a little larger.)



- Create a sketch **sketch_241120f** where a narrow rectangle whose color is taken from the image (use the center point of the rectangle) moves across the window from left to right. For an interesting effect, don't clear the background in every frame (not shown).



If you still have time, create a sketch **sketch_241120g** which is similar to #4 but contains several images, each in a different region of the drawing window.