

Homework 1

Due: 24 Jan 2024

Problem 1.1

Create a webpage in your `public_html` directory (or a subdirectory) that is at least slightly coherent, not the same as the tutorial content, and makes use of:

- header tag(s), and
- one or more of `p`, `div`, and `span`

The HTML file should be well-formed with proper headers; and it should link to a CSS file that visibly styles either the header or the text tags, or both. Finally there should be a file called `README.txt` with contents resembling the following:

To access this file from the web, use the following URL:
`http://something.blah/whatever/whatever`

with the URL fixed to be the correct one for your page.

Problem 1.2

In that same webpage or a different one, put

- an image file,
- two lists, one marked with circles and the other with squares, and
- a table with bold, centred column headers, at least one column of text (which is left-aligned), and at least one column of numbers (which is right-aligned).

For this assignment *all closable tags should be closed* even if the browser displays it ok without that. Note that the *styling* aspects of this (circle/square

markers, left/right alignment) should be done through separate CSS, not inlined, so you will need to use `class` and/or `id` attributes in the HTML so that you can coordinate that.

If these files are separate from the one for the first problem, indicate that also in the readme file and give the URL for this page as well.

Problem 1.3

There is a directory of files in the shared directory with a bunch of content (one of the readings I use when I'm teaching 121), and for this problem you'll style the files. First copy them over:

```
cd
cd public_html
mkdir hwk1
cd hwk1
cp /home/shared/210/hwk1/* .
```

(don't forget the dot on the last line, and the spaces are important!)

Then, edit the `index.html` file *only* to connect it to a CSS file that you'll write—no other edits to the HTML—and style the page to match the visual appearance shown in screenshots later in this handout (more or less). Once you have that basically working, start a *second* CSS file to match the other set of screenshots. You can leave the not-in-current-use CSS filename in a comment in the HTML file or just indicate it in the readme.

The readme file should let me know where to find the Problem 1.3 content as a URL, and also how to edit so that I can see the other CSS file (it doesn't matter which one is "current" when you hand in).

Handing in:

From your `public_html` directory, type the following:

```
handin cmsc210 hwk1 .
```

If you have other content in that directory and/or all the content for this assignment in a subdirectory, just make sure alllllll the files for this homework are in among what's handed in.

Problem 1.3 Version 1:

Representing images

Don Blaheta, Longwood University

One of the standard types of data that a modern computer needs to be able to process is the image. In the early years of digital computers, control and interaction with computers was done through numeric switches and later text terminals, and memory and computation were too expensive to be able to do much with images. Indeed, processing images at all was something of a niche area of research. Now, of course, the computer in your phone or even your wristwatch can be expected to have a graphical interface and handle images and video automatically.

As with text and other kinds of data, we will see what representational work needs to be done in order to encode an image as a sequence of numbers, so that computation can be performed.

On this page, we'll see two important representational steps: first, to take something that the human eye could physically view as a single, complex, continuous image, and break it up into small discrete pieces that can each be separately encoded; and second, to take the myriad colors that human eyes can see and divide them into separate components that can each be represented by a single number.

Subdividing images








You're probably reading this on a screen right now. Take a moment and look away from the screen, at physical things near you. There are probably a variety of straight lines in your field of vision, oriented at different angles. There are probably also a number of things that are smoothly round, whether circular or oval or some more complex shape. On all of those things there are a mix of color and shading, some shiny spots, some shadows, many with a smooth or "gradient" transition as well as many with a sharper change between colors. In order to have any chance of representing all of that visual variety, we need to break it down into smaller representable chunks.

One standard way to do so is to overlay the image to be represented with a rectangular grid. Each cell in the grid is called a *pixel*, and within this kind of system, each pixel is displayed as having a single, uniform color. Unless the underlying image is itself laid out on a rectangular grid that is perfectly aligned with the pixel grid, that means the representation is imperfect, and each pixel

Why are we working on this?
When it comes to representing information, our goal is to be able to represent any kind of information as a collection of numbers. Just numbers. That may seem particularly challenging in the case of images, but here we will work through a few aspects of that representational task to see how it works.

Skills in this section:
Subdivide images into grids of pixels;
Represent colors as RGB triples

Concepts:
Data representation

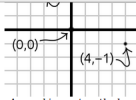
3. Measure the resolution of your laptop or smartphone in ppi.
4. The standard-definition television standard prevalent in North America through the 20th century had a resolution of 702x480. How many megapixels were there in each frame?
5. One of the current HDTV (high-definition television) standards has a resolution of 1920x1080. How many megapixels does it have in each frame?
6. Describe in English the colors represented by the following RGB triples:
 - a. (255, 255, 0)
 - b. (50, 50, 50)
 - c. (0, 0, 128)
 - d. (100, 0, 150)
 - e. (0, 128, 50)
 - f. (255, 150, 150)
 - g. (150, 100, 50)
7. Find RGB triples that match or approximate the following color swatches:
 - a. 
 - b. 
 - c. 
 - d. 
 - e. 
 - f. 
 - g. 

Credits and licensing

The Rotunda logo is property of Longwood University. Other images and all text are by Don Blaheta, licensed under a [Creative Commons BY-SA 3.0](https://creativecommons.org/licenses/by-sa/3.0/) license.

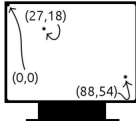
Version 2017-Jan-13 23:00

Diagram in part a of Figure 5 shows how common: the point (0,0) is right in the middle, where the axes cross; other points can be labelled, with positive or negative coordinate values that need not be nice round integers.



a: As used in most math classes

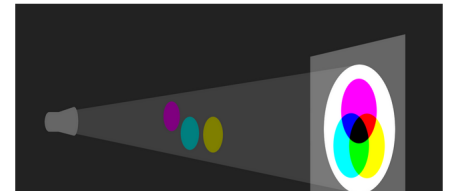
In most computer graphics contexts, though, we find it convenient to avoid fractions and negative numbers, so we put (0,0) in a corner and label each pixel separately. Furthermore, if we use the upper left corner for (0,0), with Y values increasing as we go down the image, it means that coordinates increase as we scan from top to bottom and left to right, just as if we were reading a page of text. In part b of Figure 5, you can see the origin in the upper left, and note that the other marked points have coordinates that are positive whole numbers, and indicate the number of pixels from the left or from the top of the image. It's possible to design software that doesn't work this way, of course, but nearly every image processing package you'll find that lets you refer to individual pixels will count them in this way.



b: As used in computer graphics
Figure 5: Standard Cartesian grids

Representing colors

In the last section, we showed how to break a larger image into smaller pieces, but our end goal is still to reduce everything to numbers. In order to do that, we have to think about how to represent colors, and in order to do that we need to learn a little bit about how color works.



Problem 1.3 Version 2:

Representing images

Don Blaheta, Longwood University

Why are we working on this?
 When it comes to representing information, our goal is to be able to represent any kind of information as a collection of numbers. Just numbers. That may seem particularly challenging in the case of images, but here we will work through a few aspects of that representational task to see how it works.

Skills in this section:
 Subdivide images into grids of pixels; Represent colors as RGB triples

Concepts:
 Data representation








One of the standard types of data that a modern computer needs to be able to process is the image. In the early years of digital computers, control and interaction with computers was done through numeric switches and later text terminals, and memory and computation were too expensive to be able to do much with images. Indeed, processing images at all was something of a niche area of research. Now, of course, the computer in your phone or even your wristwatch can be expected to have a graphical interface and handle images and video automatically.

As with text and other kinds of data, we will see what representational work needs to be done in order to encode an image as a sequence of numbers, so that computation can be performed.

On this page, we'll see two important representational steps: first, to take something that the human eye could physically view as a single, complex, continuous image, and break it up into small discrete pieces that can each be separately encoded; and second, to take the myriad colors that human eyes can see and divide them into separate components that can each be represented by a single number.

Subdividing images

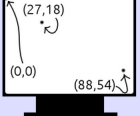
You're probably reading this on a screen right now. Take a moment and look away from the screen, at physical things near you. There are probably a variety of straight lines in your field of vision, oriented at different angles. There are probably also a number of things that are smoothly round, whether circular or oval or some more complex shape. On all of those things there are a mix of color and shading, some shiny spots, some shadows, many with a smooth or "gradient" transition as well as many with a sharper change between colors. In order to have any chance of representing all of that visual variety, we need to break it down into smaller representable chunks.

- Measure the resolution of your laptop or smartphone in ppi.
- The standard-definition television standard prevalent in North America through the 20th century had a resolution of 702x480. How many megapixels were there in each frame?
- One of the current HDTV (high-definition television) standards has a resolution of 1920x1080. How many megapixels does it have in each frame?
- Describe in English the colors represented by the following RGB triples:
 - (255, 255, 0)
 - (50, 50, 50)
 - (0, 0, 128)
 - (100, 0, 150)
 - (0, 128, 50)
 - (255, 150, 150)
 - (150, 100, 50)
- Find RGB triples that match or approximate the following color swatches:
 - 
 - 
 - 
 - 
 - 
 - 
 - 

Credits and licensing

The Rotunda logo is property of Longwood University. Other images and all text are by Don Blaheta, licensed under a [Creative Commons BY-SA 3.0](#) license.

Version 2017-Jan-13 23:00



b: As used in computer graphics
 Figure 5: Standard Cartesian grids

You've seen before, probably in a high school algebra class, the idea of using an X axis and a Y axis, perpendicular to each other, which together define a coordinate system for labelling points. The diagram in part a of Figure 5 should look familiar. The point (0,0) is right in the middle, where the axes cross; other points can be labelled, with positive or negative coordinate values that need not be nice round integers.

In most computer graphics contexts, though, we find it convenient to avoid fractions and negative numbers, so we put (0,0) in a corner and label each pixel separately. Furthermore, if we use the upper left corner for (0,0), with Y values increasing as we go down the image, it means that coordinates increase as we scan from top to bottom and left to right, just as if we were reading a page of text. In part b of Figure 5, you can see the origin in the upper left, and note that the other marked points have coordinates that are positive whole numbers, and indicate the number of pixels from the left or from the top of the image. It's possible to design software that doesn't work this way, of course, but nearly every image processing package you'll find that lets you refer to individual pixels will count them in this way.

Representing colors

In the last section, we showed how to break a larger image into smaller pieces, but our end goal is still to reduce everything to numbers. In order to do that, we have to think about how to represent colors, and in order to do that we need to learn a little bit about how color works.

