What we will cover

- Part 1: Understanding images
- Part 2: Bitmap Graphics
  - Resolution
  - Color Depth
  - Problems/conversions/calculation/etc.
- Part 3: Compression

Part 1: What is an image

- Before we can come up with a way to digitally represent an image, we need to understand what an image is.
- Then we have to break down into the smallest meaningful part, determine how many possible values there can be for each individual part, string them together and we have our image.
An Image

- To understand what an image is we need to understand human sight.
- In very general terms, what we “see” is a combination of frequencies of reflected light within a given area. (frequencies that are absorbed by a surface are not visible of course).
- A color, or shade, is just a way of describing the frequency of light waves that our eyes can detect.

Color detection

- The eye consists of two types of photoreceptors – rods and cones.
- Cones are responsible for detecting color.
- A full discussion of this topic is far beyond the scope of this class – but what we do need to know is that the cones can be broken down into red, green, and blue receptors.
- Red, green, and blue are called the primary additive colors. Remember this is in terms of LIGHT, not pigment.

Primary additive colors

- We can imagine that light is measured in terms of intensity (i.e. darkness to brightness).
- If red, green, and blue are all totally dark, the result is black (no light at all).
- If these three are of FULL intensity, the result is white (full intensity light).
- If these three are all the same intensity but somewhere in between no and full intensity, the result is a shade of gray.
If however red, green and blue light are of different intensities in relation to each other, the combined result is a non-gray color. The exact color depends on the proportion of red/green/blue light.

Of course very rarely do we perceive just a single color in a given area (unless we are looking at a large monochromatic area). The “images” we see in a given area consists of gradients of reflected light frequencies. For instance if you look at a tree you can see shades of green, brown, yellow, etc. The sky will be shades of blue. We only assign MEANING to what we see based on its context and our background knowledge. To the eye however it is just shades of light.

How many shades can the human eye detect? What do you think? There is an infinite number of shades. The conundrum here is that a computer can never represent an infinite number of anything. Unlike analog/film photographs, a digital image is limited to a discrete number of shades.
Hopefully we can understand at this point that there are two things we must consider to digitally represent an image. One is the area (size) that we wish to represent (consider “field of vision” for example). Another is the combination of colors that make up the image within the given area.

The type of computer graphics we will be studying are called Bitmaps. These are the easiest to understand. There are other types (most notably “raster” graphics) but we will not be studying these. NOTE: this discussion will only be concerned with still images (not moving images/video).

Imagine you have a large printed poster or photograph. You lay a window screen over the poster. For each individual square of the window screen, you can only choose one single color (one that comes closest “on average” to the various colors that may be inside the square). This is basically how digitizing an image is done.
Resolution

- Let’s say our screen had large holes. Each can only “hold” one color, as described. What do you think the resulting image will look like?
- Now imagine we used a different screen. It had smaller holes, but within the same area of course. Remember the same rules as before: one color per hole. Now what do you think the image will look like?
- As you can imagine, the more holes we have within a given area, the more accurate or true-to-life our image will become.
- This is because each hole will “zoom in” more closely on a specific color.
- Then we put them together. The combination of chosen colors will be a representation of the image.
- The amount of “holes” within a given area is called the resolution.

Resolution

- As you can imagine, the more holes we have within a given area, the more accurate or true-to-life our image will become.
- This is because each hole will “zoom in” more closely on a specific color.
- Then we put them together. The combination of chosen colors will be a representation of the image.
- The amount of “holes” within a given area is called the resolution.

The Pixel and Resolution

- In technical terms, each “hole” in our imaginary screen is called a picture element.
- The common abbreviation for this is the “Pixel”. (why not PicEl? 😊)
- Each pixel can only consist of a single color.
- As you can see, the “finer” the holes are within the same area, the larger the number of pixels that will make up our image.
- The number of pixels used to represent a single image is called the image’s resolution.
A digital image is 640 pixels wide by 480 pixels high.

What is its resolution in total pixels?

640 x 480 = 307200 pixels, or 307.4 K pixels.

As you probably know, digital cameras can produce digital images with resolutions in the Mega-pixel range.

You will sometimes see resolution stated in terms of A x B (area) measurements, and sometimes it will be stated in total pixels.

Our “window screen” analogy is very similar to how real image digitization equipment works. Common image digitization input devices include:

- Scanner: A printed image is “scanned” and broken into individual pixels at a specified resolution.
- Digital camera: A real life visualized image is captured in a moment in time, and broken into individual pixels at a specified resolution.

In both of these cases, remember that a particular color is also chosen for each pixel, which takes us to...

As long as we understand what a pixel is, and how pixels are used to construct a digital representation of an image, then we have to ask ourselves another question:

How many possible colors can be represented by a single pixel?

To put it another way … We already know that a single pixel can only be one color, but one out of how many possibilities?
Color depth

- Remember that, just like we break up an image into many discrete parts (pixels), we must now also determine a discrete number of shades that the image can represent.
- In other words, each pixel is really a binary pattern that represents a particular shade.
- Putting them all together gives us our image, in terms of pixels and the shade represented by each pixel.
- The number of possible shades per pixel is called the image's color depth.

Color depth

- There are several common ways that color depth is defined and used, and it depends on the type of image.
- Consider for a minute when the number of possible shades needed to "accurately" represent an image may be different, depending on the type of image.
- We will study some commonly used schemes, in varying levels of detail, in the following slides. Be aware that more schemes of varying complexity exist in the "real world" than those we will be describing.

1-bit Color depth

- In this scheme, only two possible colors can be represented (usually these are black and white). This is because a single bit can only represent one of two possible values.
- Each pixel therefore is only 1-bit "long".
- Most common applications: simple line drawings/diagrams/schematics, optical character recognition (i.e. interpreting a typewritten document).
In this scheme, only 16 possible colors can be represented. This is because a 4 bit combinations can represent one of $2^4$, or 16, possible values.

Each pixel therefore is 4–bits (or a half-byte AKA nybble) “long”.

Most common applications: very rudimentary drawings/cartoons/logos/road maps/etc. where only a small number of distinct colors is needed.

In this scheme, 256 possible colors can be represented. This is because an 8 bit combinations can represent one of $2^8$, or 256, possible values.

Each pixel therefore is one byte “long”.

Most common applications: more detailed comic/cartoon/logo/artwork/etc. In short, “drawn/painted” graphics that are not of “photographic” quality.

For the line/paint/draw type of graphics, you may be wondering what the 16 or 256 possible colors ARE?.

Like a lot of things, it again depends on the context.

Each of the patterns is “mapped” to a table of colors that the designer chooses, based on what the graphic "looks" like (which explains why the general format is called "bitmap").
For instance if I were designing a company logo that had many blue shades (for instance there was a “sky” in the background), I would make sure that various shades of blue were available to pick from.

This color “table” is stored inside the graphic file and is called a palette. Depending on the image, some will have different palette than others.

We will not concern ourselves with the real-life format of color palette in the file. Just know what they are. (NOTE: Most graphics software automatically chooses an appropriate palette based on the colors that make up an image when it is created/edited.)

The Palette

Given:
- Resolution: 800 x 600 pixels
- Color depth: 4 bit (16 colors)

How much space does the image file take up on the disk in KB?

800 x 600 pixels = 480,000 pixels
480,000 pixels * 4 bits/pixel = 1,920,000 bits.
1,920,000 bits / (8 bits/byte) = 240,000 bytes, or 240KB

Image file size: Problem 1

24-bit Color depth

In this scheme, roughly 16 million possible colors can be represented. This is because a 24 bits combinations can represent one of $2^{24}$, or 16,772,216 possible values.

Each pixel therefore is three bytes “long”.

Most common applications: photographic quality images.

We will study this scheme in more detail.
High Color depth formats

- 24-bit color is frequently referred to as "true color".
- This three-byte scheme can be better understood in terms of what the individual bytes actually represent:
  - 1st byte: $2^8$ possible intensity levels of red light
  - 2nd byte: $2^8$ possible intensity levels of green light
  - 3rd byte: $2^8$ possible intensity levels of blue light

Red/Green/Blue representation

- As you can see, each byte can represent one out of 256 possible intensity levels of red, green, or blue light.
- Remember that when we combine these intensities of R/G/B light in various combinations we get many different colors.
- So, knowing and understanding the primary additive colors of light, we were able to design a scheme that comes close to representing what we can see.

RGB format and ratios

- The three bytes of 24-bit RGB color intensity are usually represented using hexadecimal (hex) notation. Remember, two hex digits for each byte.
- Intensity goes from none 0 (00) to full 255 (in hex FF) and everything in between
- Here is pure full-intensity red: $255,0,0$ (in hex: FF0000)
- How would you describe this one? $00FF00$
Let's look at some real examples:

- Half red, half green, full blue: 128,128,255 (in hex: 8080FF)
- ¼ red, full green, ½ blue: 64, 255, 128 (in hex: 40FF80)

You can locate RGB color pickers / calculators on the web and experiment with the various combinations yourself. Try it.

Sometimes you will see a hexadecimal value written in this way:
- 0x8080FF

Sometimes you may see it written like this:
- #8080FF

Mathematically you may see it written like this:
- 8080FF

In this lesson, to avoid confusion, I will just specify “hex” or “base-16”)

I just want you to be aware of the various ways that hexadecimal values are notated, should you encounter them in other contexts.

Given:
- Resolution: 800 x 600 pixels
- Color depth: True color (24-bit)

How much space does the image file take up on the disk in MB?

800 x 600 pixels = 480,000 pixels
480,000 pixels * 24 bits/pixel = 11,520,000 bits.
11,520,000 bits / (8 bits/byte) = 1440000 bytes, or 1.44MB
Now that we know how to represent true color, and also how to calculate file sizes, here is a question:
If I have a photographic quality image, but it only contains shades of gray (like an old “black and white” photograph), can you think of a more efficient way to store this file on the disk?

The term “black and white” is commonly used in everyday language to describe images that consist of shades of gray.
- “Black and white” photographs
- “Black and white” movies
But this is not accurate when talking about digital images!
A “Black and White” image in our context means that the pixel can be either black or white (1-bit depth).
The proper term used for images consisting of shades of gray is instead “grayscale”.

Remember that with true color, each byte represents a single shade of red, green, or blue.
Remember though we also said that if the intensities of these three are identical, we will either have black, white, or an intermediate shade of gray. For example:
½ red, ½ green, ½ blue:
- 128,128,128 (in hex 808080):
Given that this is the case, how many shades of gray are possible using the 24-bit scheme?
Shades of Gray

- Each of the three bytes can represent 256 possible shades.
- But a shade of gray (including black, and white) is only produced when all three of the bytes represent an equal value!
- So there are only 256 possible shades of gray in the 24-bit scheme.
- Now if you wanted to convert an image from true color to grayscale, why waste all those extra bytes??

8-bit Grayscale format

- When a true color image is converted to grayscale, each pixel is now only one byte long.
- This is because we only need 256 possible shades of gray (from 0 to FF).
- Unlike our 8-bit Color format discussed earlier (which used different palettes mapped to individual colors), our bytes will now represent shades of gray.

Converting true color to grayscale

- We can convert true color to grayscale on a pixel–by–pixel basis.
- This is a much simplified calculation, but for our purposes … for each pixel apply the following formula:
  - Gray = \(\frac{\text{red} + \text{green} + \text{blue}}{3}\)
- \textbf{IMPORTANT}: It is “easier” to convert the bytes to base-10 numbers, perform the math, and then convert the result back to hex!
Converting color to Gray

- Example: Pixel value (in hex): 72AC30
  - 72 hex = 114 (red value)
  - AC hex = 172 (green value)
  - 30 hex = 48 (blue value)
- Gray value = \((114 + 172 + 48) / 3\)
- Gray value = 334/3
- Gray value = 111 (rounded to whole number)
- In hex this is 6F
- HELPFUL HINT: if your extended ASCII chart also has a decimal column, you can use it as a quick hex/decimal converter too!! 😊

Grayscale / image space on disk

- We started off with a true color pixel that was 3 bytes long: 72AC30
- We ended up with a grayscale pixel that is only 1 byte long: 6F
- So, if we had an image made up of true color pixels, and the image that took up about 3 MB of disk space, how much disk space would be needed for a grayscale copy of the same image?

Grayscale / image space on disk

- If each grayscale pixel is only 1/3 the size of a true color pixel, and we did not change the image size (resolution) at all, then the amount of space it will take up on the DISK is 1/3 the original amount.
- Important: Do not confuse “file size” (physical amount of space the file takes up on a disk) with the “image size” (resolution)!
Know what resolution and color depth mean
Understand the various color depth schemes and formats
Be able to calculate the FILE size given the resolution and color depth.
Be able to convert true color to grayscale.

Let’s consider modern digital cameras. Let’s say one is capable of producing 8 Megapixel images. Also assume that since these are photographs, we are using true (24-bit) color. If each pixel is then 3 bytes long, and there are 8 million of them, that is 24 million bytes, or 24MB, per photograph! That is a large file size and these images will take a very long time to upload/download if we want to use them on websites or e-mail them.

What do we do?

Image files are commonly converted to other file formats called compressed formats. Compression schemes are designed so that an image file will take up less space on a disk than it would be stored as "raw" sequences of binary digits for each pixel. There are two main types of compression schemes: lossless and lossy. Either way, the file must of course be "re-assembled" into individual pixels for use to view the image.
Lossless Compression

- Lossless compression means that no information is lost in the process.
- In other words, a file is compressed to save space. When it is opened again, the original image is re-construced exactly as it existed before the file was compressed.
- Let’s look at a common way of doing this.

Run-Length encoding

- Run-length encoding saves the image as value pairs that describe the pixel color, and how many times the same color occurs sequentially in an image.
- Consider the pixel patterns in this low color-depth (16 color / 4 bit) image
  - (note that I am showing it in bytes for clarity, so in this case each hex digit represents a single pixel!)
  - AA AA AA AA A6 66 66 33 BB BB BB BB BB
- So we can just say A,11;6,5;3,2;B,10

Run-Length encoding

- Original pixel sequence:
  - AA AA AA AA A6 66 66 33 BB BB BB BB BB
- Compressed sequence:
  - A,11;6,5;3,2;B,10
- Forgetting for a moment exactly how this is stored digitally in real life, we can see that we are not using as much space to store the image now.
- Further, when we open it up, it can easily be re-assembled to its original form by “reading” how many times the pixel needs to repeat itself.
- This compression scheme is only practical for images that have relatively low color depth (can you imagine why that is??).
- Some common lossless file types are GIF and PNG.
Lossy Compression

- Lossy compression means that in order to reduce the size of the file, certain information is deliberately removed first. (Then it can be compressed using other lossless techniques— not getting too technical here…).
- What gets discarded? The idea is to discard changes in shade and detail that the human eye may not readily detect.
- When the file is decompressed now, the information that was discarded is gone forever. It can never be restored back to its original un-compressed form.
- Note that most consumer grade digital cameras automatically perform lossy compression when saving digital photos.

JPEG Compression

- The most common form of lossy image compression is called JPEG. This stands for the “Joint Photographic Experts Group.” This is a real organization made up of smart people who came up with some complicated mathematical formulas to compress high color-depth images (like 24-bit images).
- The JPEG scheme allows us to significantly reduce the file sizes of images, allowing them to easily be uploaded/downloaded/e-mailed/stored.
- The degree of image loss vs. quality can usually be selected by whoever saves the file. The more compression, the smaller the size, but the lower the image quality.
- This is an example of the many “trade offs” that arise in systems design.

A final word to the wise...

- Hopefully you understand a little more about lossy compression than you did before.
- With that in mind … if you are working with editing/saving true-color digital images in any way, I recommend that you:
  - First save the file in a “raw” uncompressed format (common format is BMP but there are others – research them for your particular computer system).
  - Do all your editing/changing/saving/etc. ONLY in uncompressed format, each and every time.
  - Only when you are all done editing, save it as a JPG file and distribute as you see fit.
- Otherwise, you can lose more quality each time you save the file in a lossy compression format!
Image Compression: Review

- Know the difference between lossless and lossy compression.
- Know the applications for where each type of compression is most appropriate.
- Be able to reconstruct an image that was compressed using the Run–Length Encoding lossless scheme.