1. Elements and Codes

In the previous class we discussed how computers represent text and images as lists of numbers. The procedure has three parts: identify the elements, assign numbers to each element, then record that list of numbers. But what are the elements?

In an image, we said the elements are pixels. In music the elements are samples -- the height of the wave at a sequence of instants. Are these the only way to break an image or a sound into elements?

*Midi for Music*

Consider the mechanical music box. That machine does not store a song as samples of wave. Instead, it records notes for instruments. Each hole represents a note for a specific horn, bell, or drum. The sound wave is produced when those notes are played, but the holes on the cards do not encode the sound wave.

A modern equivalent is called *midi*. For midi recordings, the elements are notes. Each note has a pitch, an instrument, a duration, and some other attributes. These various components of each note are assigned numbers. Playing back a midi track is similar to the music box we saw.

Therefore, sound can be digitized by sampling a wave, or sound can be digitized by recording the sequence of notes it takes to make the noise. What sounds are suitable for midi and which ones are not?

*Basic Shapes for Graphics*

What about digitizing an image like this one:

![Image](image.png)

We can apply a grid and record a list of pixel values for the entire grid. But most of the list will be full of zero values for all the blank space. And the lines are narrow, so the pixels will be a coarse approximation. What if we take a lesson from midi and record the basic shapes rather than the individual dots? We could convert this image into a set of numbers by assigning each basic shape a code number, and then recording the position, color, and dimensions of each of the basic shapes. Let’s devise a coding system to describe this image.

There is a coding system called *svg* (for scalable vector graphics) that looks like HTML and encodes images as a set of simple graphic elements.

What kind of images are suitable for vector graphics rather than pixel (also known as *raster*) graphics? What are the advantages?

*Midi or Sample? Vector or Raster?*

Why are there two different systems for sound? Which is better? Why are there two different systems for images? Which is better?

2. How Many Bits Does It Take?

We have seen how to digitize data. We start with some text, a picture, a piece of music, and we end up with a bunch of 1’s and 0’s. How many bits does it take, and where do we put them? First, we look at how many bits it takes to represent a piece of text and then a piece of music. After that, we look at how to store those bits.

2.1. Storing Text

Consider a 10-page paper. Say the paper is double spaced and has about 12 words per line. That’s about 30 lines, each about 60 characters, so that’s about 1800 characters. With punctuation and spacing, let’s round it to 2000 characters per page. Each character takes 8 bits using ASCII, so that’s 16,000 bits. But computers actually store data in groups of 8 bits, called bytes, so that is 2000 bytes per page.

A megabyte is about 1,000,000 bytes (actually a megabyte is 1024*1024 = 1,048,576 bytes) so one megabyte can
store 1,000,000/2,000 = 500 pages of plain text. A word processing document with the extra data required for fonts, colors, boxes, etc, can add lots more, but the raw textual data is 2000 bytes per page. Therefore, a 10-page paper will use 20,000 bytes.

That means a megabyte of storage can hold 50 ten-page papers.

2.2. Storing Music

How many bytes do we need to represent a digitized song?

Here are some facts and some questions:

Facts
1. A byte is 8 bits
2. Music CDs record 44,000 y-coordinates per second, at 16 bits per sample and record left and right channels
3. A megabyte is about 1,000,000 bytes (it’s actually 1024*1024 bytes)
4. A gigabyte is about 1000 megabytes

Questions
1. How many megabytes does a 3-minute song take?
2. How much space does a 1 Gigabyte flash drive have?
3. How many 3-minute songs can a 1 Gb drive hold?

2.3. Storing Images

An image file is a long list of bits. With 24-bit color, each pixel takes 3 bytes. How many bits are needed to store a 640x480 pixel image? How big is the actual jpeg file on the disk? How do they do that?

3. Realizing Bits in Hardware

We have seen how to represent data in digital form. We express the data as a list of numbers, then we express the numbers using some number of bits per element.

For example, the number 19 is expressed in binary as 10011. Why do computers use binary, and how does a computer actually store the binary number 10011?

4. Representing Binary in the Physical World

A number in binary is a sequence of 1’s and 0’s. Humans can use a pencil or pen to make a sequence of marks on paper. Marks that look like a vertical line stand for 1’s. Marks that look like little circles stand for 0’s. Another human can look at that sequence of marks and read the number. Computers could be programmed to draw marks on paper and use cameras to read marks, but they mostly don’t. They use other systems to record and identify 1’s and 0’s. What are those systems?

Early computers, and some early music boxes we looked at, used punched cards to store and read data. A hole in a card is like a 1 -- meaning ring this bell -- and an unpunched place is like a 0 -- meaning do not ring this bell.

Any physical system that has two different states or conditions can be used to represent binary numbers. Over the years, various physical systems have been used to do so.
5. Physical Systems for Storing Bits: Punch cards, Paper Tape

The music box punch cards and the loom cards are examples of physical systems for representing 1’s and 0’s. A punch card looks like:

![Punch Card Image]

Numbers are encoded in binary by punching holes in the card. A typical card has 80 columns, one for each number. Each number can represent one character, so the card can hold up to 80 characters of text. Long sequences of text are stored on a sequence of cards.

The idea of cards was used in the Jacquard Loom of 1804-05.

**Paper Tape**

A slightly different system from the same era was paper tape. Paper tape looks like:

![Paper Tape Image]

Paper tapes were used in the telegraph industry to store messages on tape to be transmitted at speeds faster than those possible by human operators. Each hole stood for a dot or dash. The idea was modified later to store, in binary, data for computers.

Numbers are stored as holes across the tape. Each hole is a 1 or 0. Each row of holes across the tape can store one ASCII code. A long string of numbers or text is stored on a long stretch of tape.

A problem with tape and cards is that they cannot be reused. Once you punch holes in paper, you cannot unpunch them. Computers need a way to store data and then change the values - variables. How can one store 1’s and 0’s in a form that can be changed?
6. Other Media for Storing Bits

Punch cards and paper tape are easy to understand. A hole means 1, a blank means 0. What other systems do we use now to store bits? How do these systems work?

<table>
<thead>
<tr>
<th>Other Media for Storing Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEDIA</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

7. Variables - Bits in a Form You Can Change

Some of the media listed in the previous section are like punch cards and paper tape -- permanent. Once you punch holes in a card, you cannot unpunch them. But there are lots of digital media that can be changed.

If you want to revise a document or modify an image file, you need to read the bits from the disk, modify the bits, and write the revised version back to the disk. How do computers store data in variables, that is in some way that allows the values to be changed?

8. Computer Memory: Storing 1’s and 0’s

How do computers store 1’s and 0’s in a system that allows data to be stored, read, and changed? The textbook talks about the various generations of computers. The changes from one generation to the next refer to how bits are stored, read, and changed.

How can a computer store a 9x9 bitmap of the letter T? In binary, this bitmap is 111010010 without its header or color information. How can a computer store this sequence of nine 1’s and 0’s? A computer needs to use some physical quantity that can be in one of two states. Here are some possibilities:

- **Magnetism:** Magnets have a north pole and a south pole. A sequence of magnets can be used to represent 1’s and 0’s. North up means 1. Hard disks use magnetism. Core memory (see below) uses magnetism.

- **Lights:** Lights in house windows can be one or off. A dark window can mean 1, light can be 0. Early computers used spots on a TV screen to store 1’s and 0’s.

- **Buckets of water/capacitors of charge:** A bucket can hold water. A capacitor can hold charge. A full bucket can be 1, empty 0. The memory in most computers these days store charge in tiny tiny capacitors to represent 1’s. Transistors are the faucets that fill and empty these capacitors.

- **Current:** A wire can be carrying a current or not. A wire with current can be 1, a wire with no current can be 0. The little switches to turn current on and off used to be electro-mechanical relays, then vacuum tubes, and now transistors.
9. One System in Detail: Core Memory

An early solution was to use little circular magnets. A magnet can have a north pole facing one way or a north pole facing the other way. By assigning 1 to be one direction and 0 to be the other direction, we can store 1’s and 0’s on little magnets. These little magnets were called core memory and a closeup looks like:

![Core Memory Diagram](http://www.psych.usyd.edu.au/pdp-11/core.html)

Each ‘core’ had four wires passing through it. The whole group was woven into a square - like a piece of fabric with beads woven into it. By sending electricity through these wires, computers can set the magnetic direction one way, the other way, and can also see if a 1 or 0 is stored in that core.

Core memory was the major form of computer memory in the 1960’s into the first part of the 1970’s. After that, transistor-based memory became the major form of computer memory.

10. Buckets and Relays

Modern computers, like the ones on your desks and your laps, use magnetism for permanent storage on the hard disk, but use capacitors and transistors for the megabytes of RAM you use to hold variables, documents, and images when they are being processed.

![Bucket and Relays Diagram]

You could store 1’s and 0’s in a row of sinks, like those in public restrooms. To store a 1, you fill the sink, to store a 0, you leave the sink empty. To convert a 1 to a 0, you open the drain and let the water run out. To convert a 0 to a 1, you close the drain and turn on the faucet.

How do you tell if a sink is full or empty? That’s easy, you open the drain and see if water flows out. If so, there was a 1 there, but now the sink is empty, so you have to fill it again.

That’s how modern memory chips work. Each bit has a capacitor and a transistor. The capacitor holds charge, just as a sink holds water. A transistor is a valve that allows charge to flow or not. Combine a transistor and capacitor and you can store one bit.

11. Data Compression: GIF, JPEG, MP3

We have now discussed techniques for storing bits. Once people get used to storing images, movies, and sound as bits, they find there is never enough capacity to store all the swell stuff they encode as bits. They store lots of detailed images, ever more music, longer movies. How can you store data using fewer bits?

A 30Gb iPod should, based on calculations from the first part of this handout, hold about 1000 songs. But Apple claims a 30 Gb iPod can hold about 7500 songs. That’s a lot more than we computed. How do they do that?