CSE 111

Fall 2010

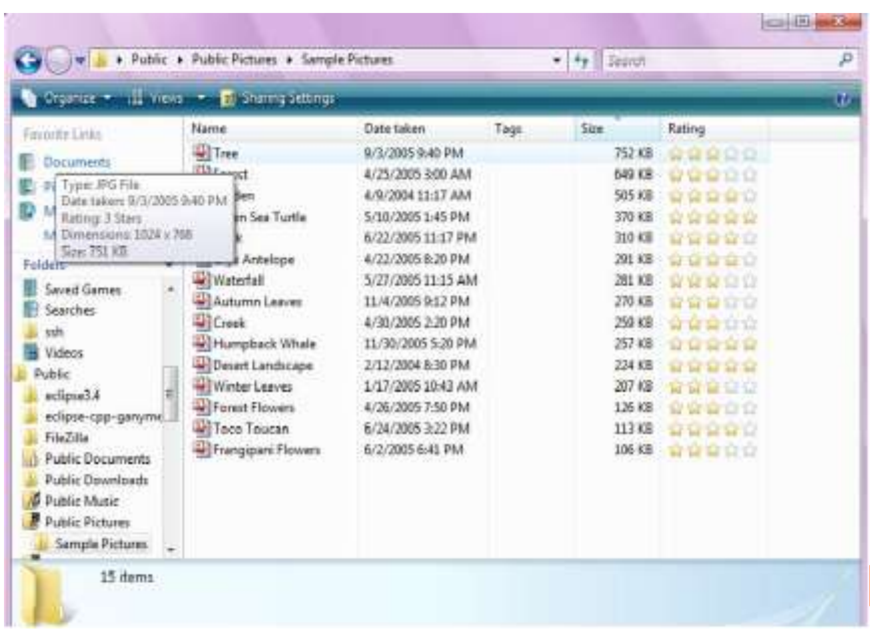
September 20 – 24

Announcements

- Lab 2 Part 1 assigned for lab sessions this week
  - Turn it in via UBLearns
- Lab 2 Part 2 next week
- Exam 1 – Monday, October 4th in lecture

Storing Image Information

- Images are made up of pixels.
- The dimensions of an image are also given in pixels.



Storing Information about Image

- Strictly Black and White Images
  - Store a 0 for no black or a 1 for black in an image
- Grayscale images
  - Store information about how much black is in an image
- Color images
  - Store three values
    - Amount of red
    - Amount of green
    - Amount of blue

Question

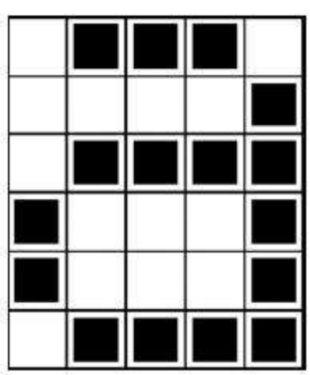
- If we use 1 byte for each red, green, and blue value, and the size of the image is 1024x768 pixels, how many bits does it take to store the information about the picture?

- 1 byte = 8 bits
- Each pixel needs 8 bits x 3 values = 24 bits
- Picture is 1024 x 768 pixels = 768432 total bits
- 768432 x 24 = 18874368 bits

Question (continued)

- 18874368 bits is how many bytes?
- 18874368 / 8 =
- 2359296 bytes
- How many kilobytes?
- 2359296 / 1024 =
- 2304 kilobytes
- How many megabytes?
- 2304 / 1024 =
- 2.25 megabytes

Sending Image Information



1, 3, 1
4, 1
1, 4
0, 1, 3, 1
0, 1, 3, 1
1, 4

Problems with Encoding Images

- Size
  - They take up a lot of bits
- Scaling
  - "Zooming" in on a part of the images causes it to distort

Alternative Method

- Store information about the lines and shapes inside the image instead about the values of the actual pixels.
  - This is how True Type fonts work.
  - Small version small
  - Small version not so small

Sound Representation

- Major challenge about encoding sound…

Sound Encoding

- Sound is encoded by sampling the sound amplitudes and encoding the amplitudes.
- Long ago (telephone technology – NOT cell phone technology)
  - 8000 samples per second
- Music needs more
  - 44,100 samples per second
  - 16 bits per sample (32 if you want stereo)

QUESTION

- How many bits to store 1 second of sound?
- 44,100 x 16 = 675600 bits(mono) 1351200 (stereo)

- One minute of sound?
- 1351200 x 60 = 1351200860 bits

- Three minutes of sound?
- 1351200860 x 3 = 4053602580 bits
- = 506700322.5 bytes = 494824.53 KB = 483.22 MB = 0.47 GB

SOUND ENCODING

- MIDI (Musical Instrument Digital Interface)
  - Records the note that is played and the duration the note is played for.
  - Example: Guitar plays the note C for 3 seconds
    - Uses 4 bytes of storage for that information
- Sort of an electronic encoding of sheet music

Compression

- First decision:
  - Are you willing to lose information?

Image Compression

- JPEG is a lossy compression

loses information when compressing

How Can it Be Okay to Lose Information?

- JPEG relies on the human eye's limitations in regards to sight
- Human eye is not nearly as sensitive as a computer
- Example (from a website)
  - 000000 is the encoding for the color black
  - 000001 is a different color

JPEG Compression

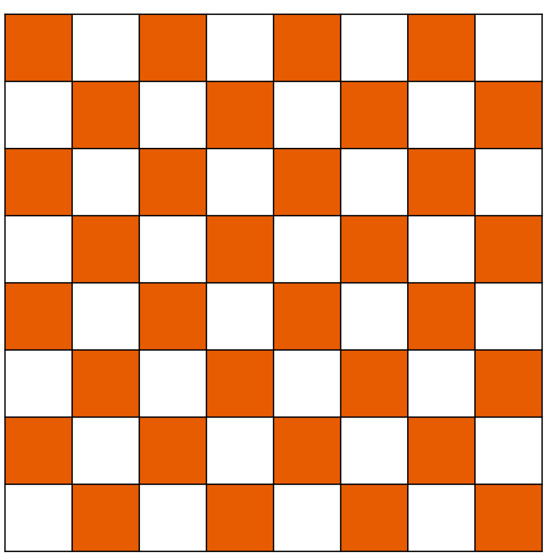
- Step 1: Average the chrominance in the picture in 2x2 squares
- What is the reduction in size?

Reduced by a factor of 4

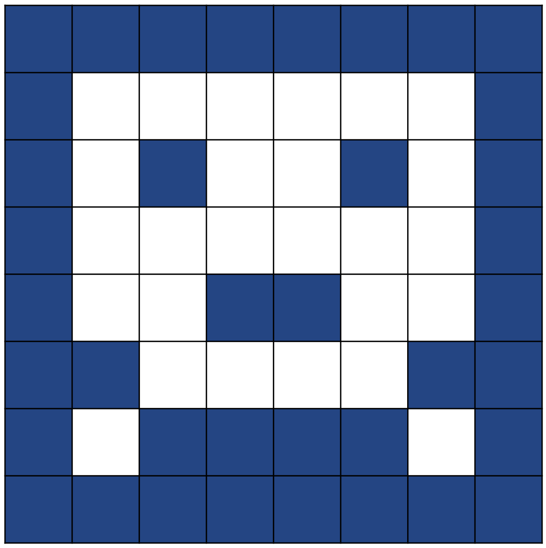
JPEG Compression

- Step 2: Divide image into 8x8 pixel blocks.
  - These blocks are analyzed and information is stored about how the pixels relate to each other, not actual color information.

Example



Another Example



JPEG Compression

- Step 3: Use
  - Run-length encoding
  - Variable-length encoding
  - Relative encoding

- Total compression of at least a factor of 10, sometimes as much as 30

Other Compression Techniques

- Run-length encoding

Find repeating elements & store information about the repeats

Ex) 4-1's, 2 0's, 5 1's, 1 0's, 2 1's,

Variable Length Encoding

Not all elements encoded use the same number of bits

- More frequent characters use shorter codes

Variable Length Encoding

- Example text:
- How did we get here? ← 20 characters × 8 bits = 160 bits

d - 2	H - 1	t - 1
e - 4	i - 1	w - 2
g - 1	o - 1	? - 1
h - 1	r - 1	Space - 4

Variable Length Encoding

- Encodings
- d - 000
- e - 110
- g - 0100
- h - 0101
- H - 0110
- i - 0111
- o - 1000
- r - 1001
- t - 1010
- w - 001
- ? - 1011
- space - 111

Variable Length Encoding

- For you later:
- How many bits does it take to encode the message with the encodings I've given on the previous slide?
- What does the encoded message look like?

Relative Encoding

Records the differences between segments of the code

Compressing Movies

- Movies/videos are shot in frames
  - 24, 25, 50, 60, 120 frames per second
- From frame to frame in a film, how much does the image change?
- When we want to compress movies, we store the entirety of certain frames, and then store the changes between the completely stored frames.
  - 1 frame completely stored for every 15 not completely stored

Compressing Sound

- Takes advantage of the limitations of the human ear to hear certain sounds along with the other compression techniques we have discussed previously.

Base 8 Numbers

- Called octal numbers (remember we had binary and decimal numbers already)
- Octal numbers use the digits 0,1,2,3,4,5,6,7 to form numbers

Converting from Octal to Decimal

- Base 8 number: 45

8^1 8^0

$$(4 \times 8^1) + (5 \times 8^0)$$

$$(4 \times 8) + (5 \times 1)$$

$$32 + 5 = 37$$

Converting from Decimal to Octal

- Decimal number: 62

Base 8 number: 76

$8\overline{)62}$ = 7 R6; 62 − 56 = 6

$8\overline{)7}$ = 0 R7

Convert from Base 8 to Base 2

- First, convert from Base 8 to Base 10
- Then, convert from Base 10 to Base 2

Base 16 Numbers

- Called hexadecimal numbers
- Hexadecimal numbers use the digits 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F to form numbers
-

Converting from Hexidecimal to Decimal

- Base 16 number EC

16^1 16^0

$(E \times 16^1) + (C \times 16^0)$

$(14 \times 16^1) + (12 \times 16^0)$

$(14 \times 16) + (12 \times 1)$

$224 + 12 = 236$

A=10
B=11
C=12
D=13
E=14
F=15

Converting from Decimal to Hexidecimal

- Base 10 number: 135

16 | 135 = 8 R 7 (135 − 128 = 7)

16 | 8 = 0 R 8 (8 − 0 = 8)

Base 16 number: 87

Extra Problems

- The following are octal numbers, convert them to decimal numbers:
  - 345
  - 4678
  - 23
  - 777
- The following are hexidecimal numbers, convert them to decimal numbers:
  - A34
  - 56FF
  - EDEC
  - C5

Extra Problems

- The following are decimal numbers, convert them to octal numbers and hexidecimal numbers.
  - 1254
  - 347
  - 24
  - 6739

Error Correction

- Data transfers are not always perfect.

- Can we build into our encodings a way to tell if there was an error while the information was being transferred?

Parity Bits

- If we add a parity bit to our encoding, we can use it to help us check for errors.

Parity bit: Extra bit added to our encoding.
The value of the bit will be set so that the entire encoding will have an even or odd number of 1's

Hamming Distance

- See description in text

ISBN Numbers

- 10-digit version

1 1 1 1 1 1 1 1 1 1

Group (Language sharing country group)

publisher

title

Check bit (Checks to see if the number is a valid ISBN)

ISBN Numbers

- 13-digit version

1 1 1 1 1 1 1 1 1 1 1 1 1

EAN number (indicates industry)

Group (Language sharing country group)

publisher

title

Check bit (Checks to see if the number is a valid ISBN)

How Does the Checking Work?

- ISBN-10
- $x_{10} = (1x_1+2x_2+3x_3+4x_4+5x_5+6x_6+7x_7+8x_8+9x_9) \bmod 11$

What is mod?

- The operation mod returns the remainder from the division.
- For example, if you ask for 7 mod 3, you would divide 3 into 7 and the answer is the remainder from the division (1 in this case).

How Does the Checking Work?

- ISBN-10
- $x_{10} = (1x_1+2x_2+3x_3+4x_4+5x_5+6x_6+7x_7+8x_8+9x_9) \bmod 11$

- Example: 0-321-52403-9

$$x_{10} = ((1*0)+(2*3)+(3*2)+(4*1)+(5*5)+(6*2)+(7*4)+(8*0)+(9*3)) \bmod 11$$

$$x_{10} = (0+6+6+4+25+12+28+0+27) \bmod 11$$

$$x_{10} = 108 \bmod 11$$

$$x_{10} = 9$$

$$11 \overline{)108} = 9 \text{ R } 9 \quad (108 - 99 = 9)$$

ISBN-13 Check Digit

- Take each digit left to right and alternate multiplying by 1 & 3. Sum the products and then do mod 10 on that sum. Subtract that answer from 10 and that is the check digit.

ISBN-13 Check Digit

- Example: 978-0-321-52403-4

$(9*1)+(7*3)+(8*1)+(0*3)+(3*1)+(2*3)+(1*1)+$
$(5*3)+(2*1)+(4*3)+(0*1)+(3*3)$

$9+21+8+0+3+6+1+15+2+12+0+9=86$

8R6
10 | 86
−80
6

$86 \bmod 10 = 6$

$10-6=4$ is the check digit

Arithmetic with Binary Numbers

- Addition
- First we need to remember 4 basic facts

$$\begin{array}{r} 0 \\ \underline{+0} \\ 0 \end{array} \qquad \begin{array}{r} 0 \\ \underline{+1} \\ 1 \end{array} \qquad \begin{array}{r} 1 \\ \underline{+0} \\ 1 \end{array} \qquad \begin{array}{r} 1 \\ \underline{+1} \\ 10 \end{array}$$

Binary Addition

```
1 1 1   1
  101101
+  11001
 1000110
```

Subtraction

5 - 3

5 + (-3)