Digital Information

INFO/CSE 100, Spring 2005

Fluency in Information Technology

http://www.cs.washington.edu/100

Readings and References

- Reading
  - » *Fluency with Information Technology*
    - Chapter 11, Representing Multimedia Digitally
- Wikipedia - The Free Encyclopedia
  - » Arabic numerals, ASCII
    - http://en.wikipedia.org/wiki/Arabic_numerals
    - http://en.wikipedia.org/wiki/Ascii
- Cyrillic Text
    - http://www.dimka.com/ru/cyrillic/

Info Representation

- Adult humans have 32 teeth
  - » sometimes a tooth or two is missing!
- How can we represent a set of teeth?
  - » How many different items of information?
    - 2 items - *tooth* or *no tooth*
  - » How many "digits" or positions to use?
    - 32 positions - one per tooth socket
  - » Choose a set of symbols

no tooth: 0 *tooth*: 1

What's your tooth number?

incisors		canines	pre-molars		molars		
0000	0000	0000	0000	0000	0000	0000	0000

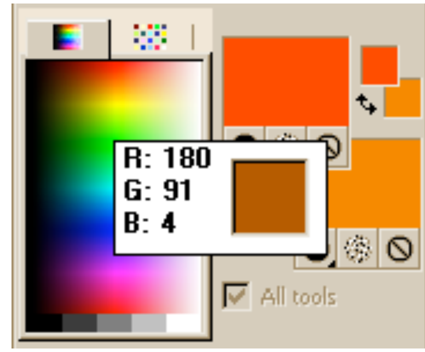
no teeth ↔ 0000 0000 0000 0000 0000 0000 0000 0000

1111	1111	1111	1111	1111	0000	0000	0000

no molars ↔ 1111 1111 1111 1111 1111 0000 0000 0000

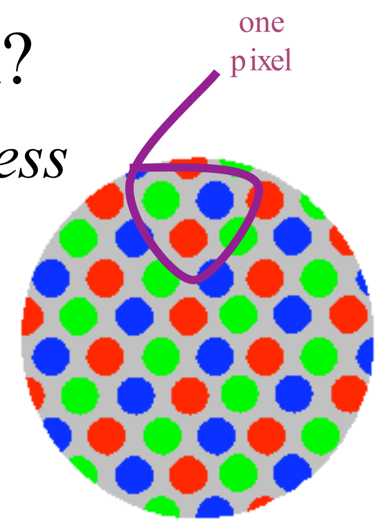
How many possible combinations? $2 \times 2 \times 2 \times 2 \times \ldots \times 2 = 2^{32} \approx$ 4 Billion

Info Representation



- Color monitors combine light from Red, Green, and Blue phosphors to show us colors
- How can we represent a particular color?
  - » How many different items of information?
    - 256 items - *distinguish 256 levels of brightness*
  - » How many "digits" or positions to use?
    - 3 positions - *one Red, one Green, one Blue*
  - » Choose a set of symbols

    brightness level represented by the numbers 0 to 255

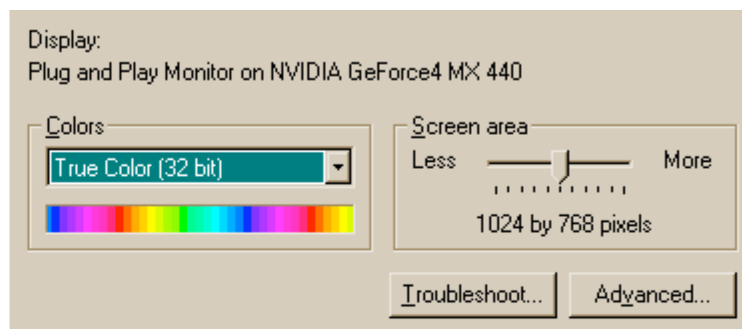


What is the pixel's color?

	red	green	blue
	255	0	0
	255	102	0
	128	128	128
	0	0	0

How many possible combinations?
$256 \times 256 \times 256 = 256^3 \approx 16$ Million

16 M colors is often called "True Color"



How can we store numbers?

- We want to store numbers
  - » 0 to 255 for color brightness
  - » 0 to 4B for tooth configuration
  - » 0 to 255 for ASCII character codes
- What do we have available in memory?
  - » *Binary digits*
    - 0 *or* 1
    - on *or* off
    - clockwise *or* counter-clockwise

0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1

…

The hardware is binary

- 0 and 1 are the only symbols the computer can actually store directly in memory
  - » a single bit is either *off* or *on*
- How many numbers can we represent with 0 and 1?
  - » How many different items of information?
    - 2 items - *off* or *on*
  - » How many "digits" or positions to use?
    - let's think about that on the next slide
  - » Choose a set of symbols
    - already chosen: 0 and 1

How many positions should we use?

It depends: how many numbers do we need?

one position

0	two numbers
1	

two positions

0	0	four numbers
0	1	
1	0	
1	1	

three positions

0	0	0	eight numbers
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

The sky's the limit

- We can get as many numbers as we need by allocating enough positions
  - » each additional position means that we get *twice* as many values because we can represent *two* numbers in each position
  - » these are *base 2* or *binary* numbers
    - each position can represent two different values
- How many different numbers can we represent in base 2 using 4 positions?

1010100101010101110000101010101001111010010101011 …

How can we read binary numbers?

Let's look at the equivalent *decimal* (ie, *base 10*) numbers.

binary base 2		decimal base 10
0	⇔	0
1		1

binary base 2			decimal base 10
0	0	⇔	0
0	1		1
1	0		2
1	1		3

binary base 2				decimal base 10
0	0	0	⇔	0
0	0	1		1
0	1	0		2
0	1	1		3
1	0	0		4
1	0	1		5
1	1	0		6
1	1	1		7

111_2 represents *exactly the same quantity* as 7_{10}
They are just different ways of representing the same number.

Position matters!

binary base 2				⇔	decimal base 10
	0	0	0		0
	0	0	1		1
	0	1	0		2
	0	1	1		3
	1	0	0		4
	1	0	1		5
	1	1	0		6
	1	1	1		7
1	0	0	0		

position represents _	position represents **4**	position represents **2**	position represents **1**	base 10
1	0	1	0	base 2

What do the positions represent?

			$2\times2\times2\times2$	$2\times2\times2$	2×2	2	1	
$2^7 = 128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	base 10
1	0	0	0	1	0	1	0	base 2

Each position represents one more multiplication by the base value. For binary numbers, the base value is 2, so each new column represents a multiplication by 2.

What base 10 decimal value is equivalent to the base 2 binary value 10001010_2 shown above?

Some Examples

$2^7 = \mathbf{128}$	$2^6 = \mathbf{64}$	$2^5 = \mathbf{32}$	$2^4 = \mathbf{16}$	$2^3 = \mathbf{8}$	$2^2 = \mathbf{4}$	$2^1 = \mathbf{2}$	$2^0 = \mathbf{1}$	base 10
_	_	_	_	_	_	_	_	base 2

$$10_2 = 2_{10}$$

$$100_2 = 4_{10}$$

$$110_2 = 4_{10} + 2_{10} = 6_{10}$$

$$111_2 = 4_{10} + 2_{10} + 1_{10} = 7_{10}$$

$$1000_2 = 8_{10}$$

$$1001_2 = 8_{10} + 1_{10} = 9_{10}$$

Recall: What do number positions represent?

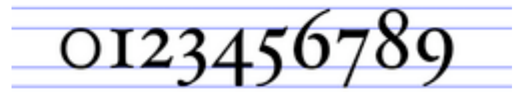
$2^7 = 128$	$2^6 = 64$	$2^5 = 32$	$2\times2\times2\times2$ $2^4 = 16$	$2\times2\times2$ $2^3 = 8$	2×2 $2^2 = 4$	2 $2^1 = 2$	1 $2^0 = 1$	base 10
1	0	0	0	1	0	1	0	base 2

$$1 \cdot 128 + 1 \cdot 8 + 1 \cdot 2 = 138_{10}$$

Each position represents one more multiplication by the base value.

For binary numbers, the base value is 2, so each new column represents a multiplication by 2.

Use the base, Luke



- Each position represents one more multiplication by the base value
  - » The base value can be **2** - *binary numbers*
    - Two symbols: 0 and 1
    - Each column represents a multiplication by two
  - » The base value can be **10** - *decimal numbers*
    - Ten symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    - Each column represents a multiplication by ten

$10\times10\times10$	10×10	10	1	
$10^3 = 1000$	$10^2 = 100$	$10^1 = 10$	$10^0 = 1$	base 10
0	1	3	8	base10

$$1 \cdot 100 + 3 \cdot 10 + 8 \cdot 1 = 138_{10}$$

text figure image is from Wikipedia

Base 16 Hexadecimal

- The base value can be **16** - *hexadecimal numbers*
  - » Sixteen symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - » Each column represents a multiplication by sixteen
  - » Hex is easier to use than binary because the numbers are shorter even though *they represent the same value*

16×16×16	16×16	16	1	
$16^3 = 4096$	$16^2 = 256$	$16^1 = 16$	$16^0 = 1$	base 10
0	0	8	A	base 16

$$8 \cdot 16 + 10 \cdot 1 = 138_{10}$$

Four binary bits ⇔ One hex digit

binary base 2					hexdecimal base 16		decimal base 10
0	0	0	0		0		0
0	0	0	1		1		1
0	0	1	0		2		2
0	0	1	1	⇔	3	⇔	3
0	1	0	0		4		4
0	1	0	1		5		5
0	1	1	0		6		6
0	1	1	1		7		7

binary base 2					hexdecimal base 16		decimal base 10
1	0	0	0		8		8
1	0	0	1		9		9
1	0	1	0		A		10
1	0	1	1	⇔	B	⇔	11
1	1	0	0		C		12
1	1	0	1		D		13
1	1	1	0		E		14
1	1	1	1		F		15

Binary to Hex examples

$$\underbrace{1\,0\,0\,0}_{8}\;\underbrace{0\,0\,1\,0}_{2}\;\underbrace{0\,0\,0\,0}_{0}\;\underbrace{0\,1\,1\,1}_{7}\;\underbrace{1\,0\,1\,0}_{A}\;\underbrace{0\,0\,0\,1}_{1}\;\underbrace{0\,0\,0\,0}_{0}\;\underbrace{1\,1\,1\,1}_{F}$$

base 2

base 16

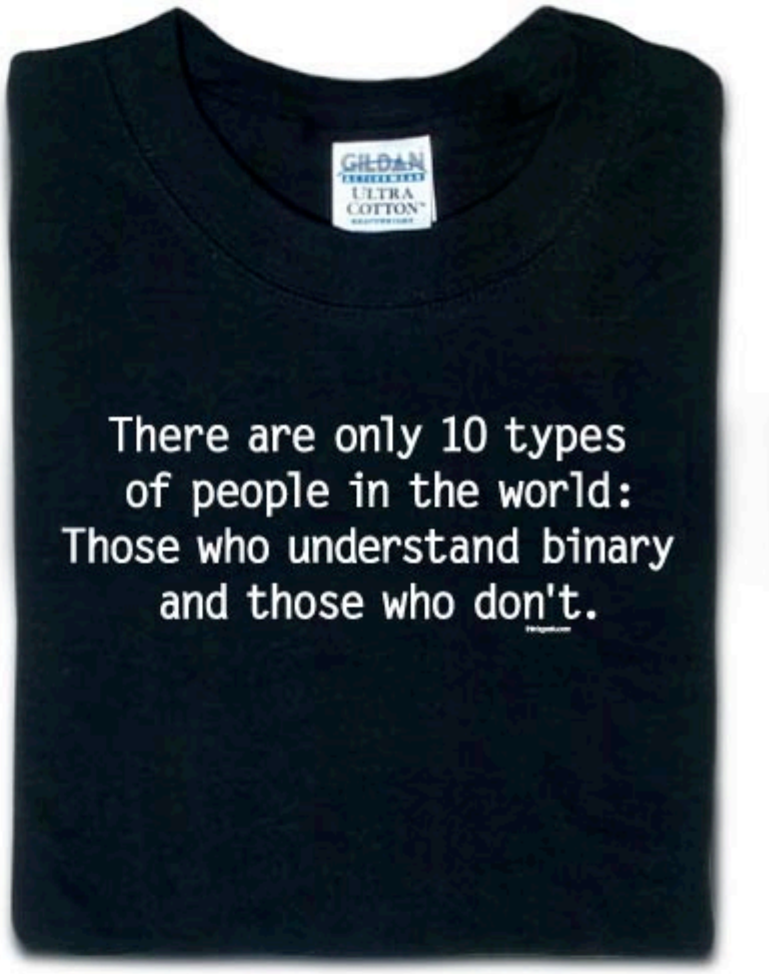
$$10000010000001111010000100001111_2 = 8207A10F_{16}$$

$$1\,0\,0\,0\;\;0\,0\,1\,1\;\;0\,1\,0\,0\;\;0\,1\,0\,1\;\;0\,1\,1\,0\;\;1\,0\,0\,1\;\;1\,0\,1\,1\;\;1\,1\,1\,0$$

base 2

$$10000011010001010110100110111110_2 = \underline{\hspace{4cm}}_{16}$$

Whew! We are now official geeks ...



http://www.thinkgeek.com/tshirts/frustrations/5aa9/

Recall: The hardware is binary

- How many numbers can we represent with 0 and 1?
  - » As many as we want, it just takes a little more space to get a bigger range
- So what can we represent with these numbers?
  - » Anything that has a numeric value or can be associated with a numeric value
  - » Number of people, index into a list, account balance, ...
  - » Alphabetic characters, punctuation marks, display tags
  - » Any signal that can be converted into numeric values
    - colors, sounds, water level, blood pressure, temperature
  - » Computer instructions

Represent numbers

- How many bit positions to allocate?
  - » Depends on the desired range
  - » 8 bits → 0 to 255
    - or -128 to +127
  - » 16 bits → 0 to 65535
    - or -32768 to +32767
  - » 32 bits → 0 to 4294967296
    - or -2B to +2B

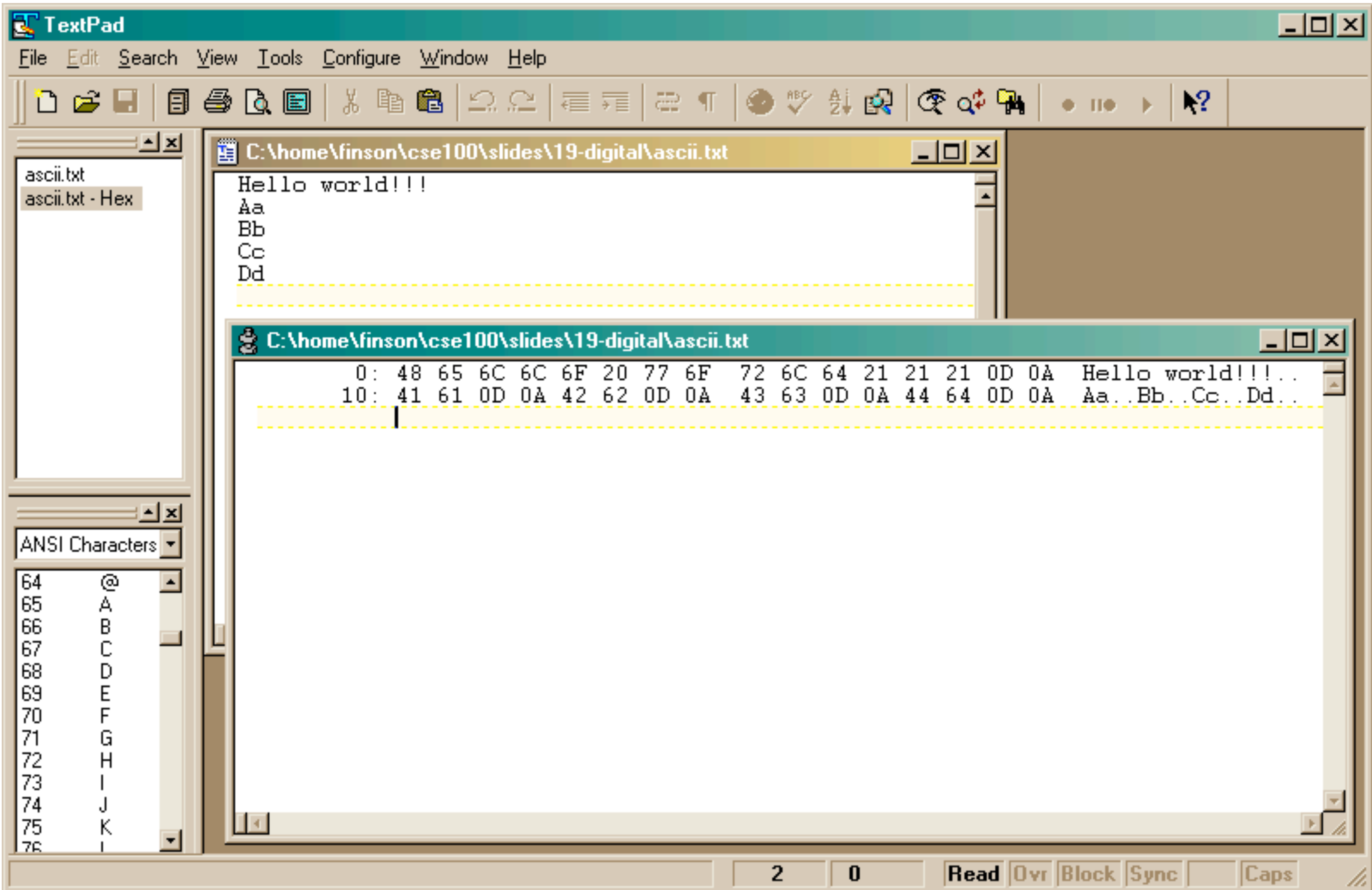
Represent Text - ASCII

- Assign a unique number to each character
  - » 7-bit ASCII
    - Range is 0 to 127 giving 128 possible values
    - There are 95 printable characters
    - There are 33 control codes like tab and carriage return

```
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
'abcdefghijklmno
pqrstuvwxyz{|}~
```

imageis from Wikipedia

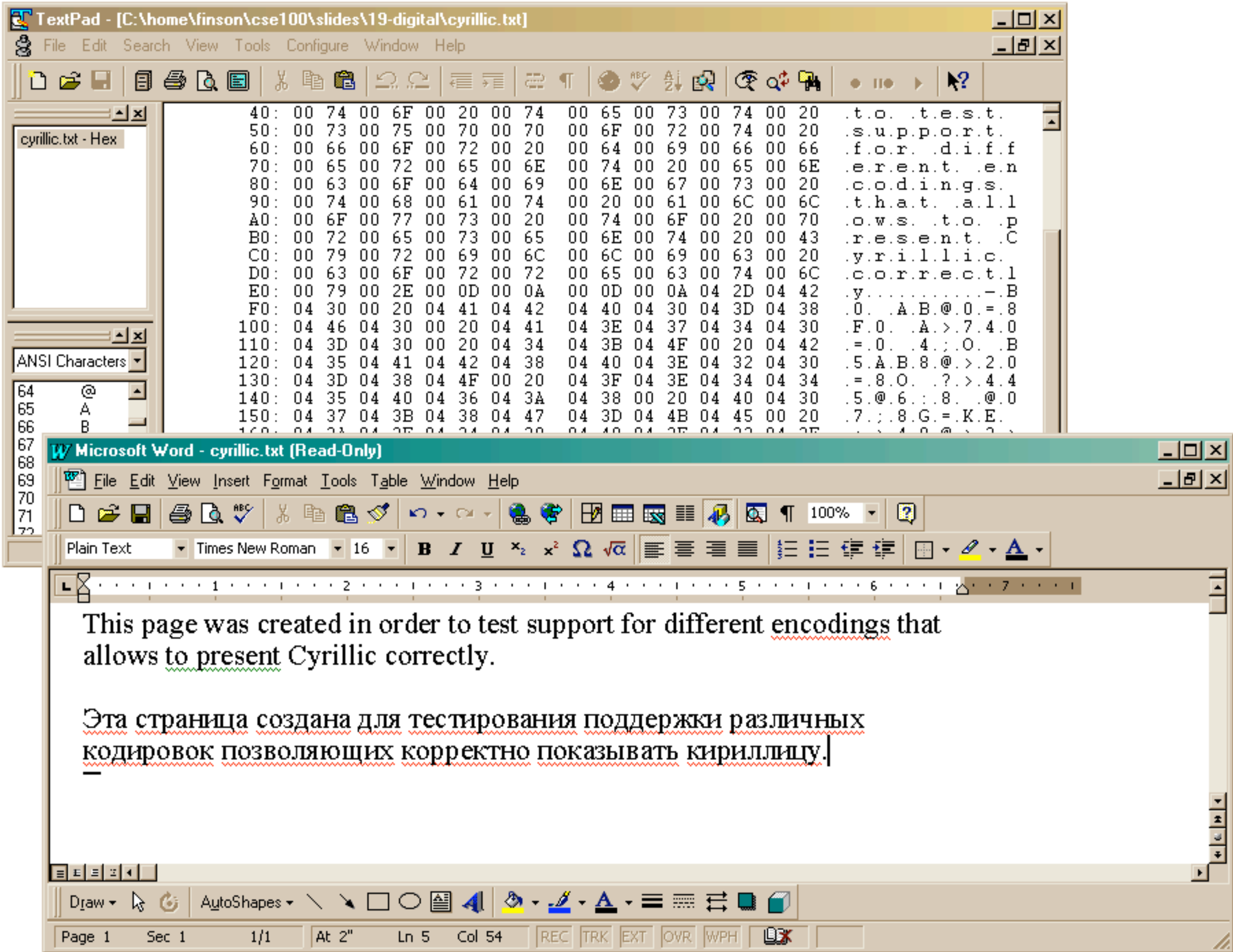
ASCII text



TextPad
File Edit Search View Tools Configure Window Help
ascii.txt
ascii.txt - Hex
C:\home\finson\cse100\slides\19-digital\ascii.txt
Hello world!!!
Aa
Bb
Cc
Dd
C:\home\finson\cse100\slides\19-digital\ascii.txt
0: 48 65 6C 6C 6F 20 77 6F 72 6C 64 21 21 21 0D 0A Hello world!!!..
10: 41 61 0D 0A 42 62 0D 0A 43 63 0D 0A 44 64 0D 0A Aa..Bb..Cc..Dd..
ANSI Characters
64 @
65 A
66 B
67 C
68 D
69 E
70 F
71 G
72 H
73 I
74 J
75 K
2
0
Read Ovr Block Sync Caps

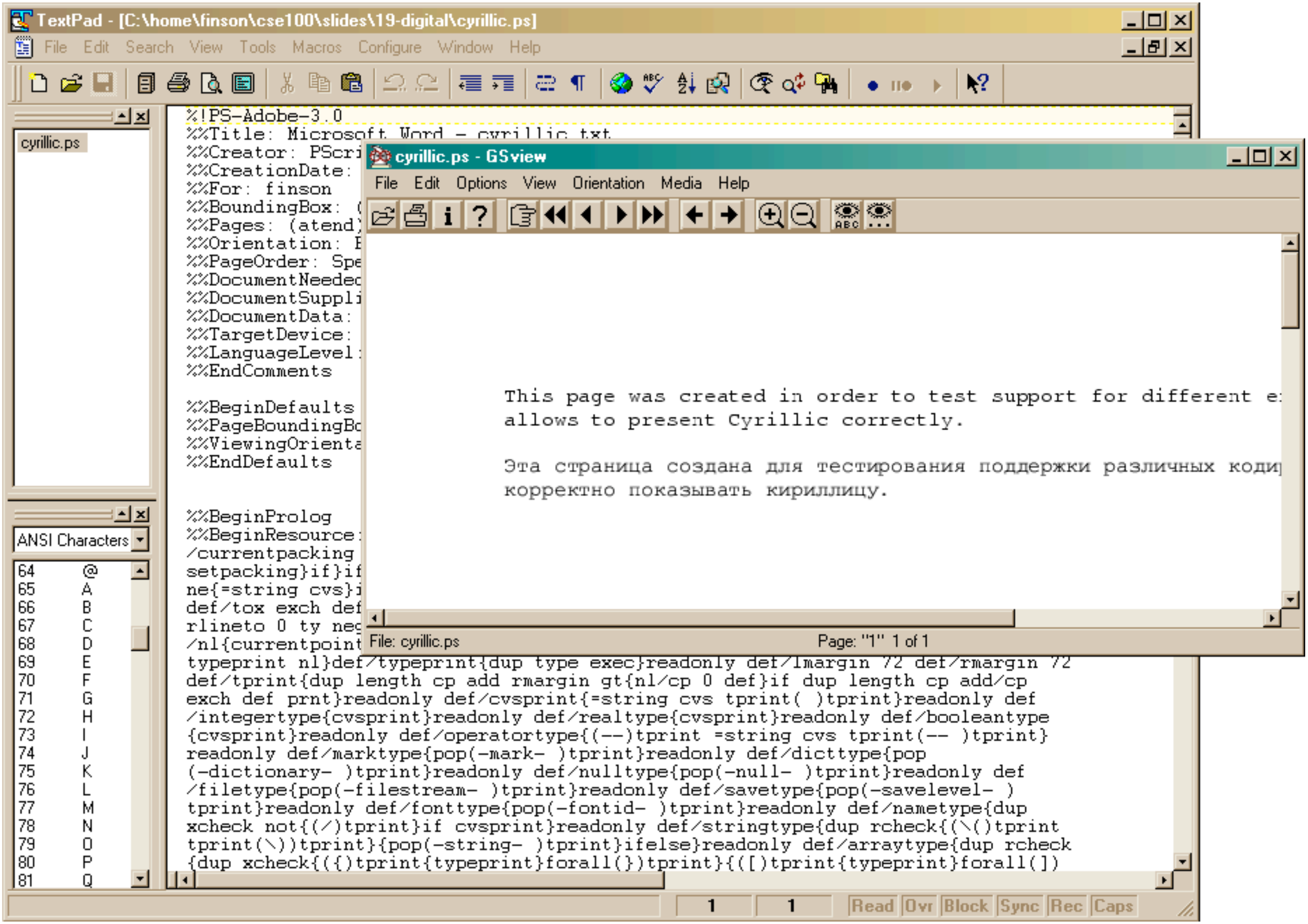
Represent Text - Unicode

- The goal of Unicode is to provide the means to encode the text of every document people want to store in computers
- Unicode aims to provide a unique number for each letter, without regard to typographic variations used by printers
- Unicode encodes each character in a number
  - » the number can be 7, 8, 16, or 32 bits long
  - » 16-bit encoding is common today


TextPad - [C:\home\finson\cse100\slides\19-digital\cyrillic.txt]
File Edit Search View Tools Configure Window Help
cyrillic.txt - Hex
40: 00 74 00 6F 00 20 00 74 00 65 00 73 00 74 00 20 .t.o. .t.e.s.t.
50: 00 73 00 75 00 70 00 70 00 6F 00 72 00 74 00 20 .s.u.p.p.o.r.t.
60: 00 66 00 6F 00 72 00 20 00 64 00 69 00 66 00 66 .f.o.r. .d.i.f.f
70: 00 65 00 72 00 65 00 6E 00 74 00 20 00 65 00 6E .e.r.e.n.t. .e.n
80: 00 63 00 6F 00 64 00 69 00 6E 00 67 00 73 00 20 .c.o.d.i.n.g.s.
90: 00 74 00 68 00 61 00 74 00 20 00 61 00 6C 00 6C .t.h.a.t. .a.l.l
A0: 00 6F 00 77 00 73 00 20 00 74 00 6F 00 20 00 70 .o.w.s. .t.o. .p
B0: 00 72 00 65 00 73 00 65 00 6E 00 74 00 20 00 43 .r.e.s.e.n.t. .C
C0: 00 79 00 72 00 69 00 6C 00 6C 00 69 00 63 00 20 .y.r.i.l.l.i.c.
D0: 00 63 00 6F 00 72 00 72 00 65 00 63 00 74 00 6C .c.o.r.r.e.c.t.l
E0: 00 79 00 2E 00 0D 00 0A 00 0D 00 0A 04 2D 04 42 .y.........-.B
F0: 04 30 00 20 04 41 04 42 04 40 04 30 04 3D 04 38 .0. .A.B.@.0.=.8
100: 04 46 04 30 00 20 04 41 04 3E 04 37 04 34 04 30 .F.0. .A.>.7.4.0
110: 04 3D 04 30 00 20 04 34 04 3B 04 4F 00 20 04 42 .=.0. .4.;.O. .B
120: 04 35 04 41 04 42 04 38 04 40 04 3E 04 32 04 30 .5.A.B.8.@.>.2.0
130: 04 3D 04 38 04 4F 00 20 04 3F 04 3E 04 34 04 34 .=.8.O. .?.>.4.4
140: 04 35 04 40 04 36 04 3A 04 38 00 20 04 40 04 30 .5.@.6.:.8. .@.0
150: 04 37 04 3B 04 38 04 47 04 3D 04 4B 04 45 00 20 .7.;.8.G.=.K.E.
ANSI Characters
Microsoft Word - cyrillic.txt (Read-Only)
File Edit View Insert Format Tools Table Window Help
Plain Text
Times New Roman
16
This page was created in order to test support for different encodings that allows to present Cyrillic correctly.
Эта страница создана для тестирования поддержки различных кодировок позволяющих корректно показывать кириллицу.
Page 1 Sec 1 1/1 At 2" Ln 5 Col 54

Represent Text - Postscript

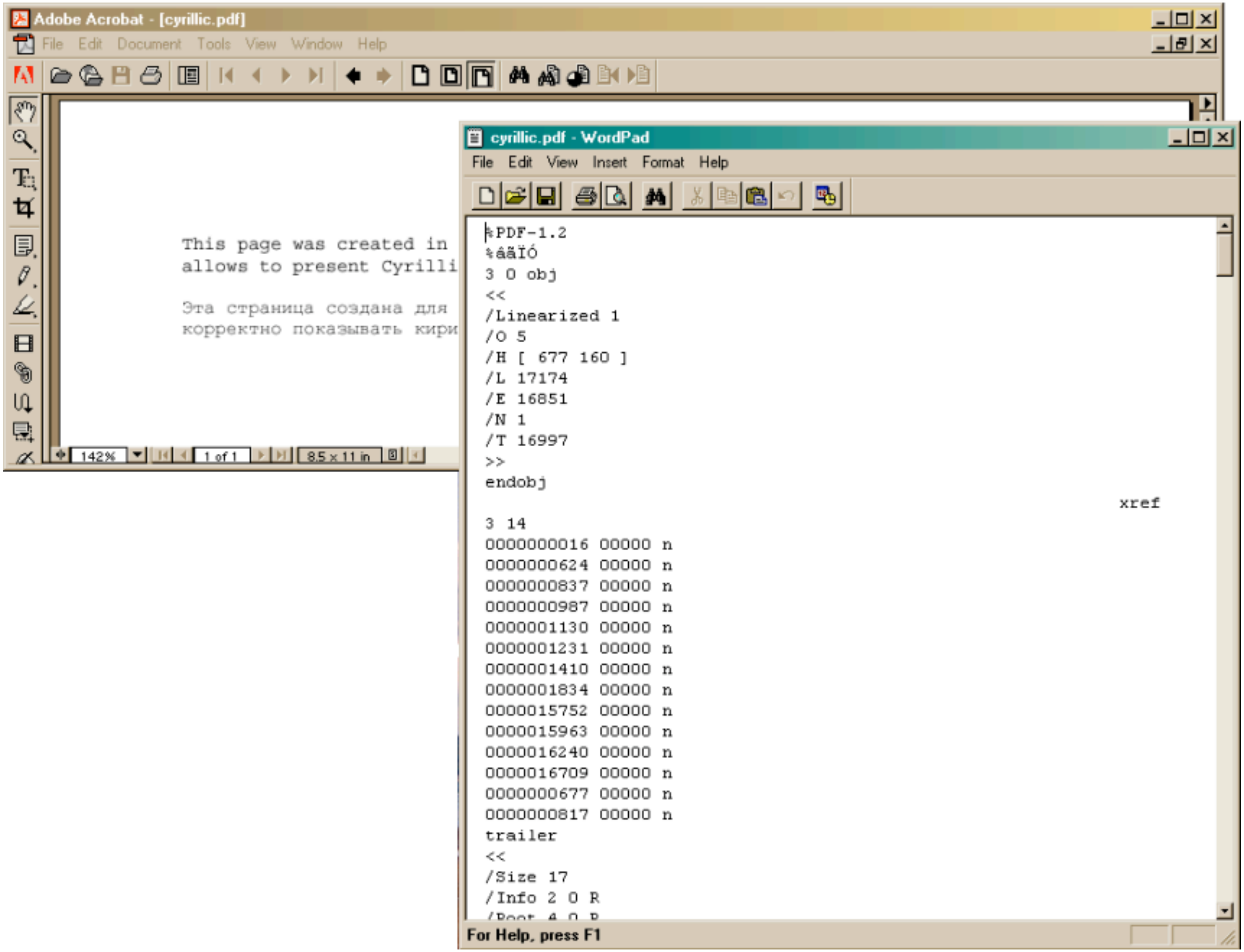
- Postscript is a page description language somewhat like HTML
  - » The file is mostly text and can be looked at with a regular text editor
  - » programs that know what it is can interpret the embedded commands
  - » Programs *and printers* that understand Postscript format can display complex text and graphical images in a standard fashion



TextPad - [C:\home\finson\cse100\slides\19-digital\cyrillic.ps]
File Edit Search View Tools Macros Configure Window Help
cyrillic.ps
ANSI Characters
%!PS-Adobe-3.0
%%Title: Microsoft Word - cyrillic.txt
%%For: finson
%%EndComments
%%BeginDefaults
%%EndDefaults
%%BeginProlog
typeprint nl}def/typeprint{dup type exec}readonly def/lmargin 72 def/rmargin 72
def/tprint{dup length cp add rmargin gt{nl/cp 0 def}if dup length cp add/cp
exch def prnt}readonly def/cvsprint{=string cvs tprint( )tprint}readonly def
/integertype{cvsprint}readonly def/realtype{cvsprint}readonly def/booleantype
{cvsprint}readonly def/operatortype{(--)tprint =string cvs tprint(--)tprint}
readonly def/marktype{pop(-mark-)tprint}readonly def/dicttype{pop
(-dictionary-)tprint}readonly def/nulltype{pop(-null-)tprint}readonly def
/filetype{pop(-filestream-)tprint}readonly def/savetype{pop(-savelevel-)
tprint}readonly def/fonttype{pop(-fontid-)tprint}readonly def/nametype{dup
xcheck not{(/)tprint}if cvsprint}readonly def/stringtype{dup rcheck{(\()tprint
tprint(\))tprint}{pop(-string-)tprint}ifelse}readonly def/arraytype{dup rcheck
{dup xcheck{({)tprint{typeprint}forall(})tprint}{([)tprint{typeprint}forall(])
cyrillic.ps - GSview
File Edit Options View Orientation Media Help
This page was created in order to test support for different e
allows to present Cyrillic correctly.
Эта страница создана для тестирования поддержки различных коди
корректно показывать кириллицу.
File: cyrillic.ps
Page: "1" 1 of 1
Read Ovr Block Sync Rec Caps

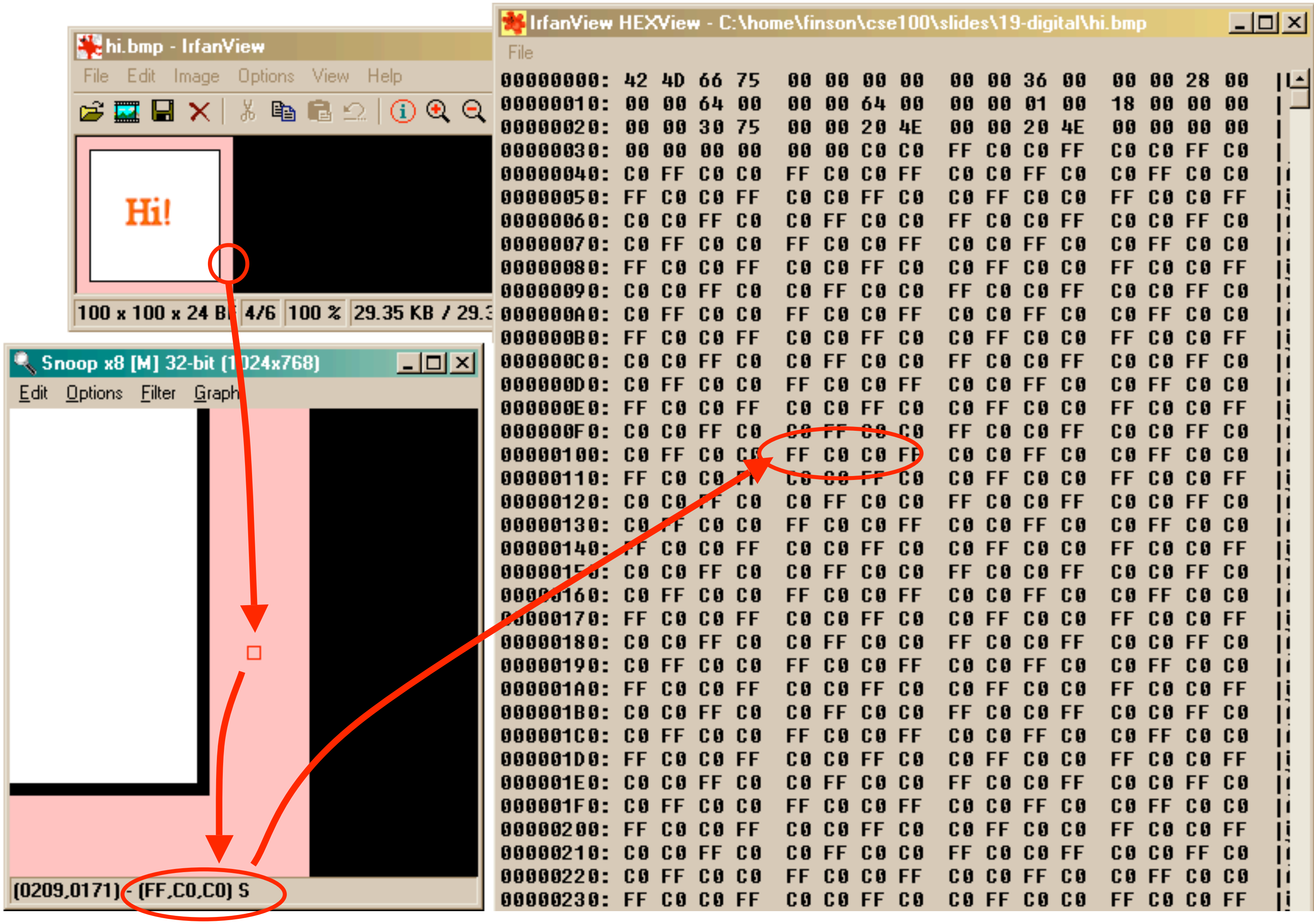
Represent Text - PDF

- PDF is another page description language based on Postscript
- The file is mostly text
  - » can be looked at with a regular text editor
  - » programs that know what it is can interpret the embedded commands
  - » just like Postscript and HTML in that respect


Adobe Acrobat - [cyrillic.pdf]
File Edit Document Tools View Window Help
This page was created in
allows to present Cyrilli
Эта страница создана для
корректно показывать кири
142%
1 of 1
8.5 x 11 in
cyrillic.pdf - WordPad
File Edit View Insert Format Help
%PDF-1.2
%âãÏÓ
3 0 obj
<<
/Linearized 1
/O 5
/H [677 160]
/L 17174
/E 16851
/N 1
/T 16997
>>
endobj
xref
3 14
0000000016 00000 n
0000000624 00000 n
0000000837 00000 n
0000000987 00000 n
0000001130 00000 n
0000001231 00000 n
0000001410 00000 n
0000001834 00000 n
0000015752 00000 n
0000015963 00000 n
0000016240 00000 n
0000016709 00000 n
0000000677 00000 n
0000000817 00000 n
trailer
<<
/Size 17
/Info 2 0 R
For Help, press F1

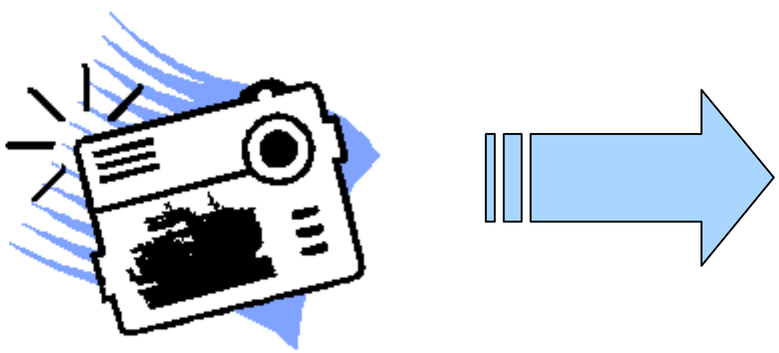
Represent Color - Bit Map

- Numbers can represent anything we want
- Recall that we can represent colors with three values
  - » Red, Green, Blue brightness values
- There are *numerous* formats for image files
  - » All of them store some sort of numeric representation of the brightness of each color at each pixel of the image
  - » commonly use 0 to 255 range (or 0 to FF_{16})

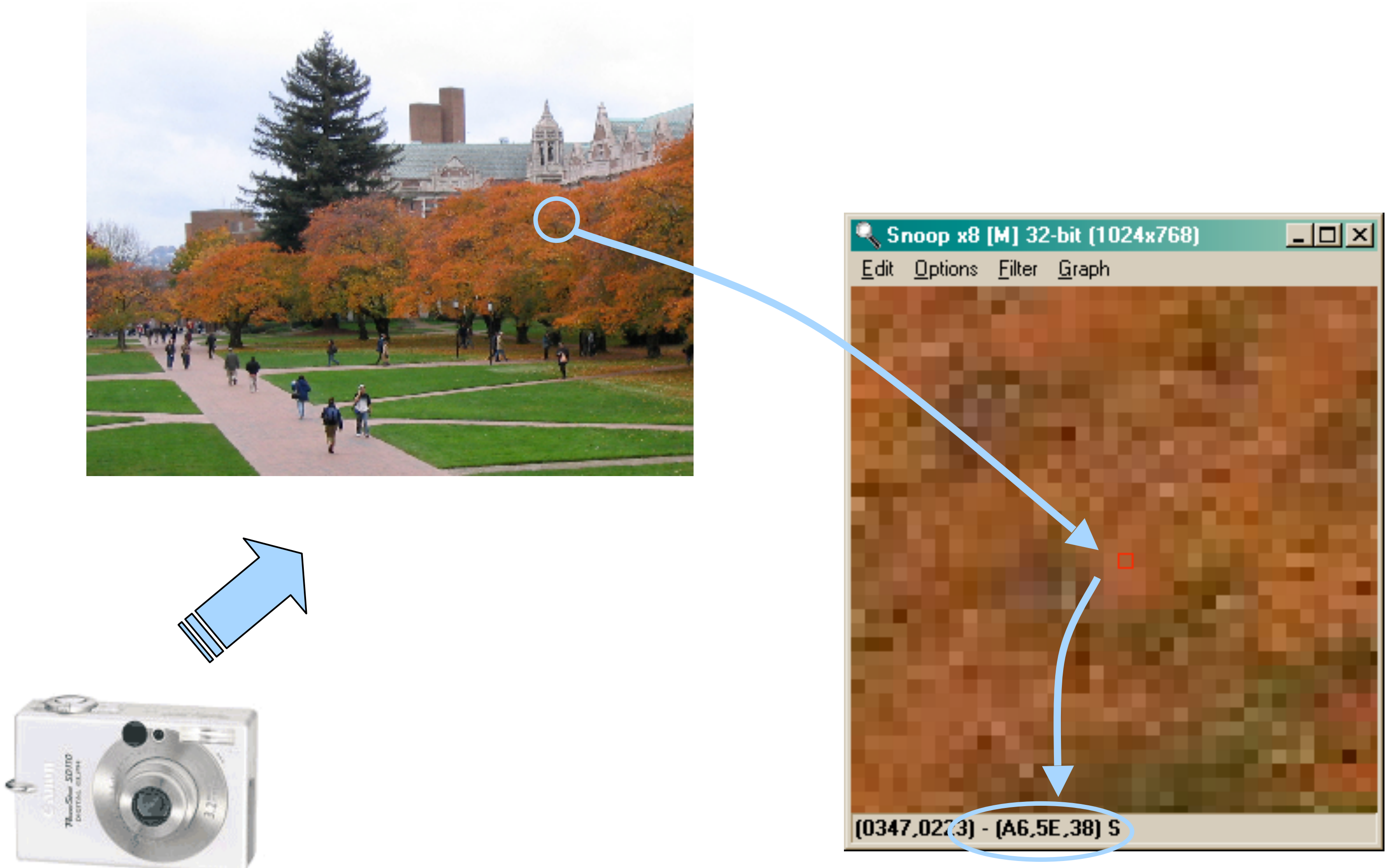

hi.bmp - IrfanView
File Edit Image Options View Help
Hi!
100 x 100 x 24 B
4/6
100 %
29.35 KB / 29.3
Snoop x8 [M] 32-bit (1024x768)
Edit Options Filter Graph
(0209,0171) - (FF,C0,C0) S
IrfanView HEXView - C:\home\finson\cse100\slides\19-digital\hi.bmp
File
00000000: 42 4D 66 75 00 00 00 00 00 00 36 00 00 00 28 00
00000010: 00 00 64 00 00 00 64 00 00 00 01 00 18 00 00 00
00000020: 00 00 30 75 00 00 20 4E 00 00 20 4E 00 00 00 00
00000030: 00 00 00 00 00 00 C0 C0 FF C0 C0 FF C0 C0 FF C0
00000040: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000050: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
00000060: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
00000070: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000080: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
00000090: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
000000A0: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
000000B0: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
000000C0: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
000000D0: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
000000E0: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
000000F0: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
00000100: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000110: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
00000120: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
00000130: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000140: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
00000150: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
00000160: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000170: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
00000180: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
00000190: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
000001A0: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
000001B0: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
000001C0: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
000001D0: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
000001E0: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
000001F0: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000200: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
00000210: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
00000220: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000230: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF

What about "continuous" signals?

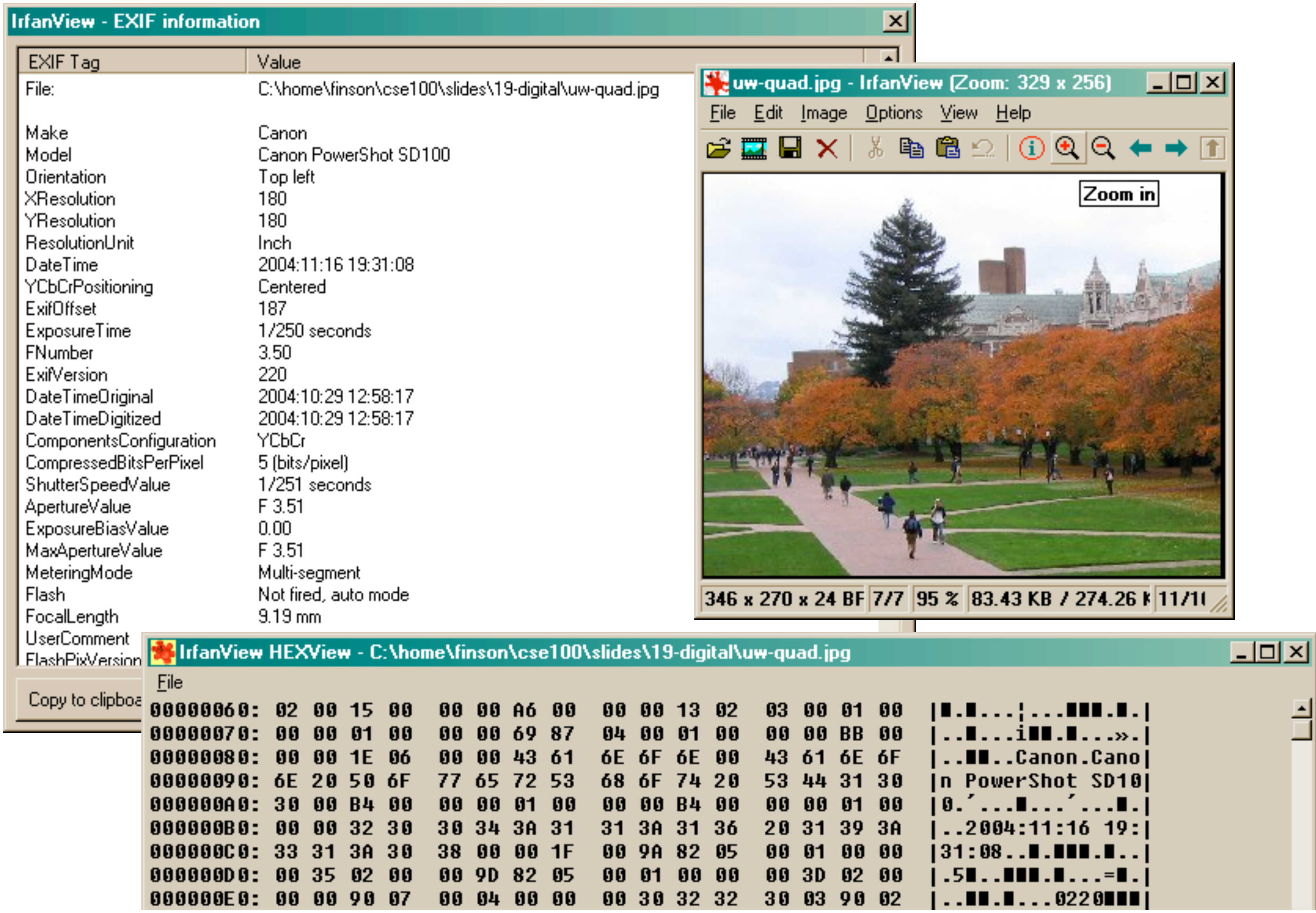
- Color and sound are natural quantities that don't come in nice discrete numeric quantities
- But we can “make it so!”



Digitized image contains color data



And much, much more!



Summary

- Bits can represent any information
  - » Discrete information is directly encoded using binary
  - » Continuous information is made discrete
- We can look at the bits in different ways
  - » The format guides us in how to interpret it
  - » Different interpretations let us work with the data in different ways