Lecture C1
Data Representation

Computing and Art : Nature, Power, and Limits
CC 3.12: Fall 2007

Functionalia

Instructor

Chipp Jansen, ***chipp@sci.brooklyn.cuny.edu***

Course Web Page

`http://www.sci.brooklyn.cuny.edu/~chipp/cc3.12/`

- **HW B** (Both Parts 1 and 2) DUE Wednesday Oct. 3, 11:59 pm
- **Midterm 1** - Monday Oct. 15th - *in-class*
  - Covers Unit **A**, Unit **B**, and Unit **C**

Office Hours | Extra Help

My Office Hours :

Mondays 12:30 to 1:30 - basement Ingersoll 0317N

Bridges Student Resource Center : 0317N

Monday	1:30 to 5pm
Tuesday	1:30 to 3:30
Wednesday	1:30 to 3:30
Thursday	12:45 - 1:45, 2:15 - 3:15

Functionalia

Today:

- Dynamic HTML (JavaScript)
- Run-time vs. Compile-time
- Data Representation

Making Web-pages Dynamic

Interactivity in Web-pages

- links (the anchor tag -

- `<a href="pagetolinkto.html"> Click here </a>`

- image maps - use images as interfaces

Dynamic content (giving the user control)

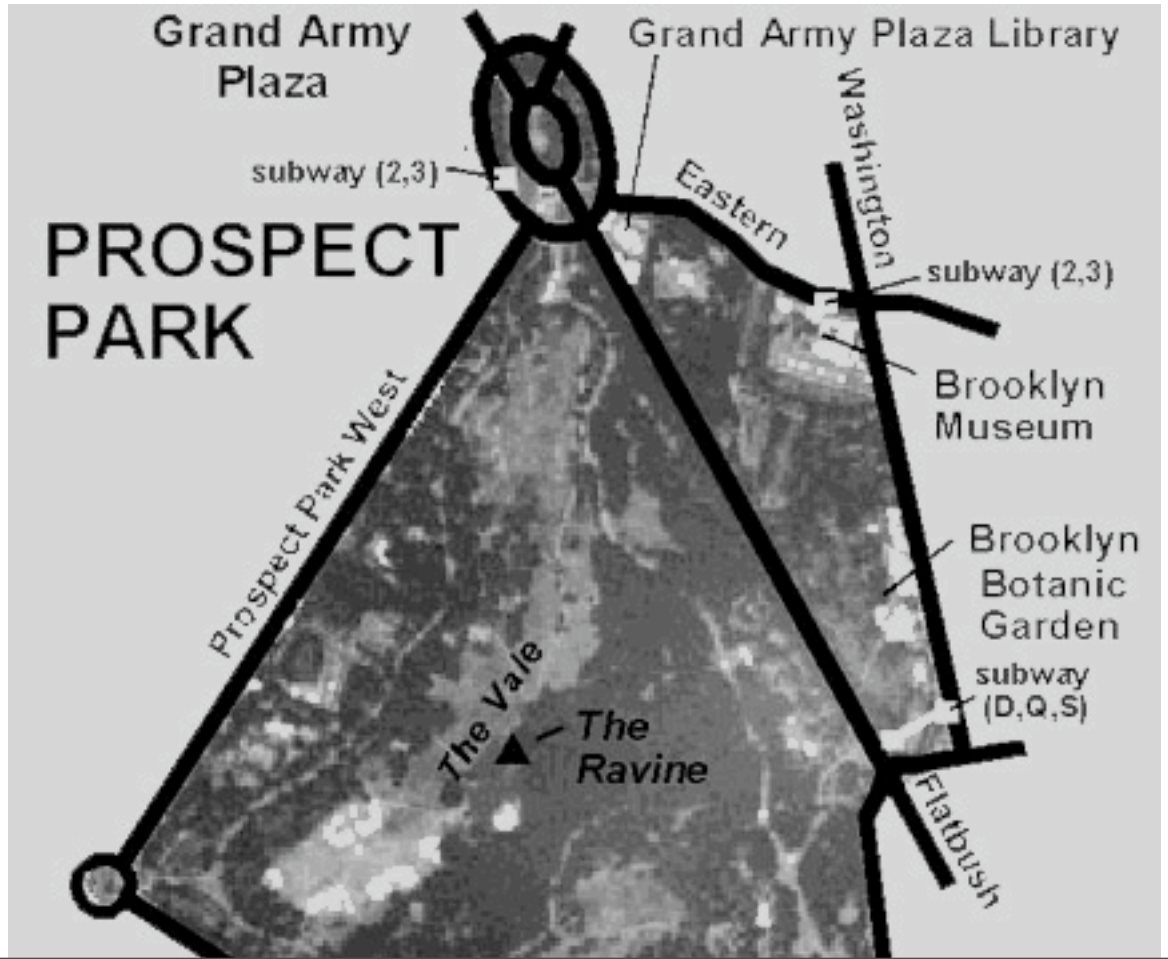
- Forms

- Javascript

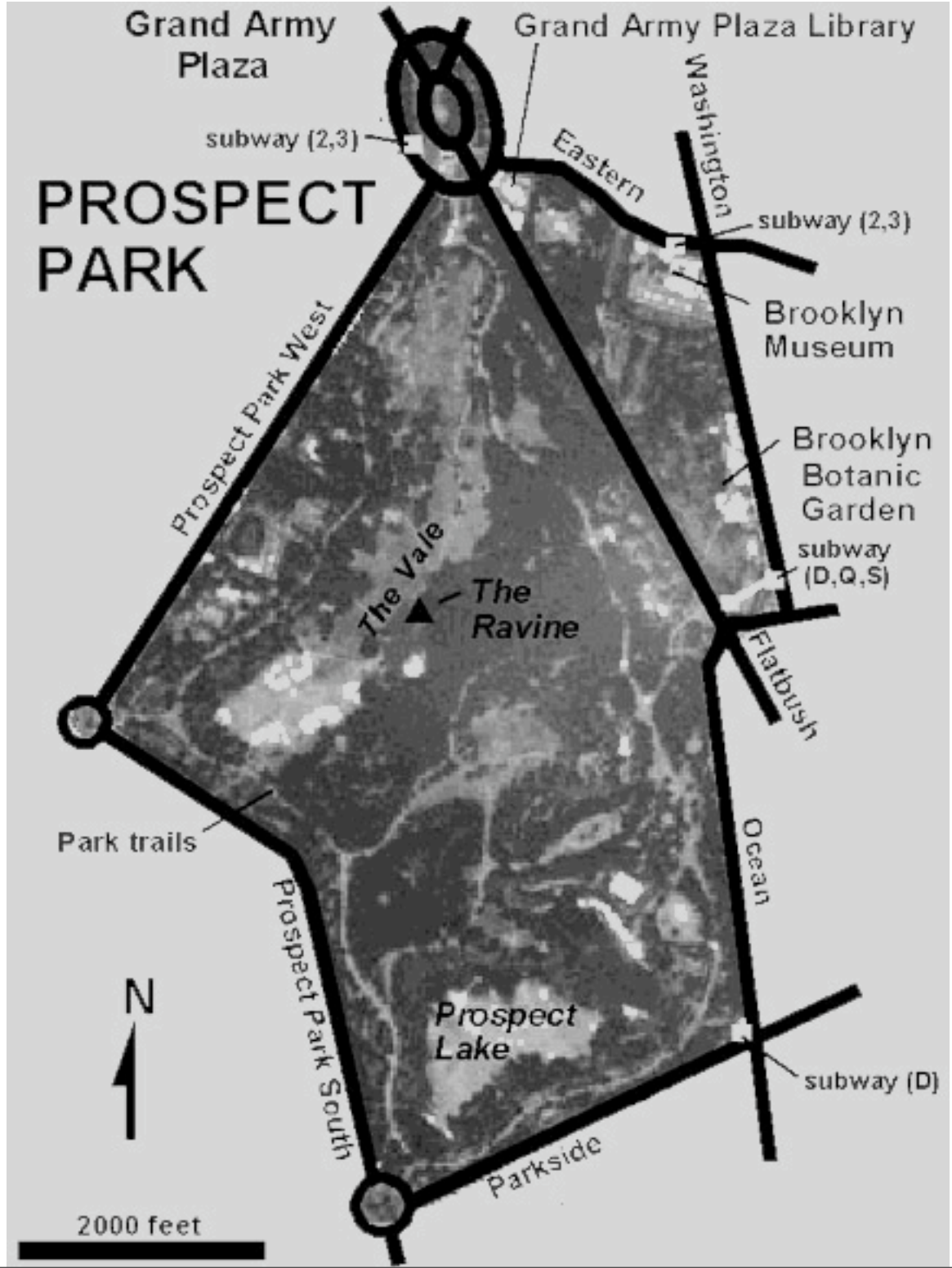
HTML : Image Maps

- Can specify areas of an image as having links (remember a tags)

- Use images as interfaces.

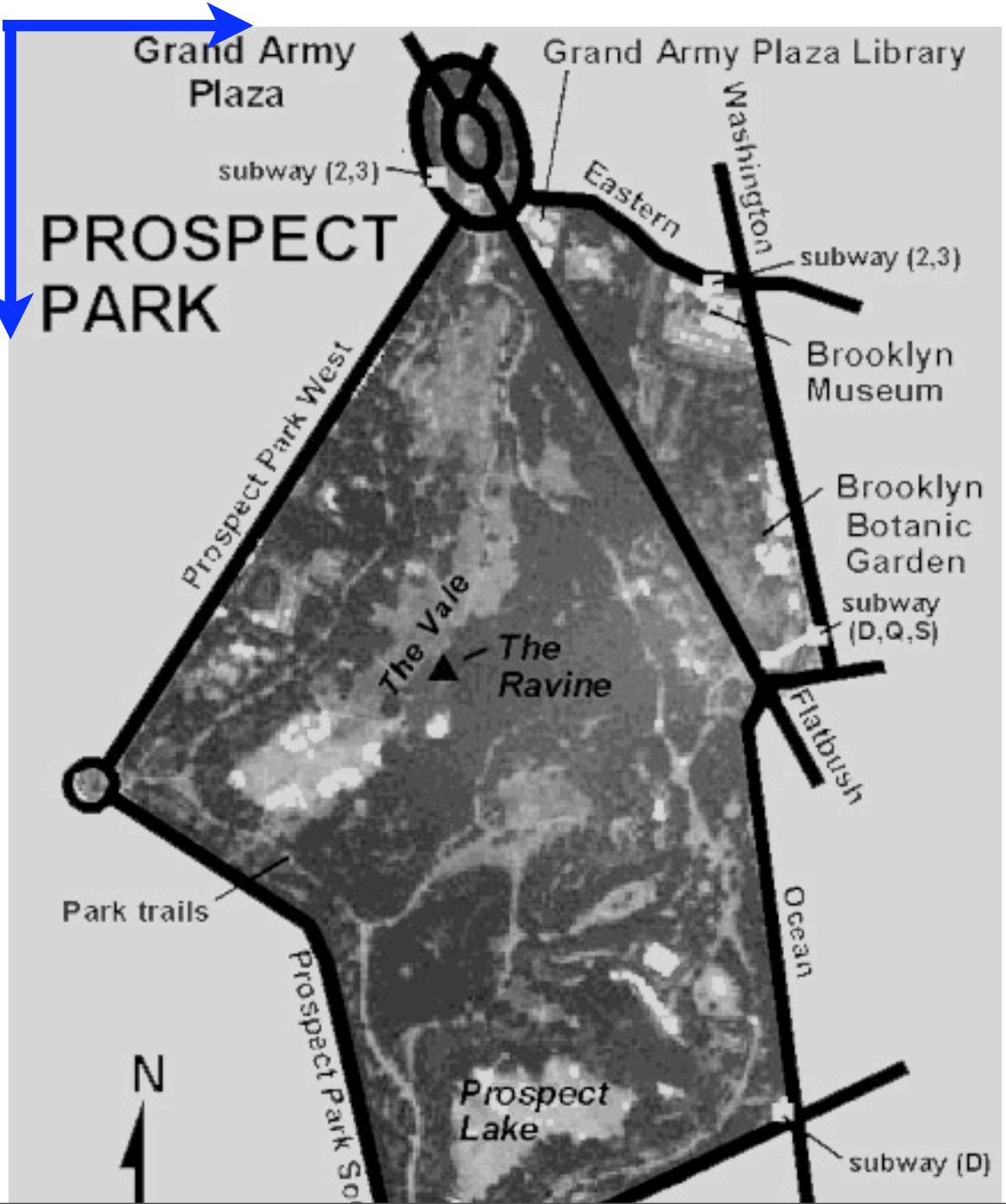


Origin is at the upper lefthand corner pixel.



x direction

y direction

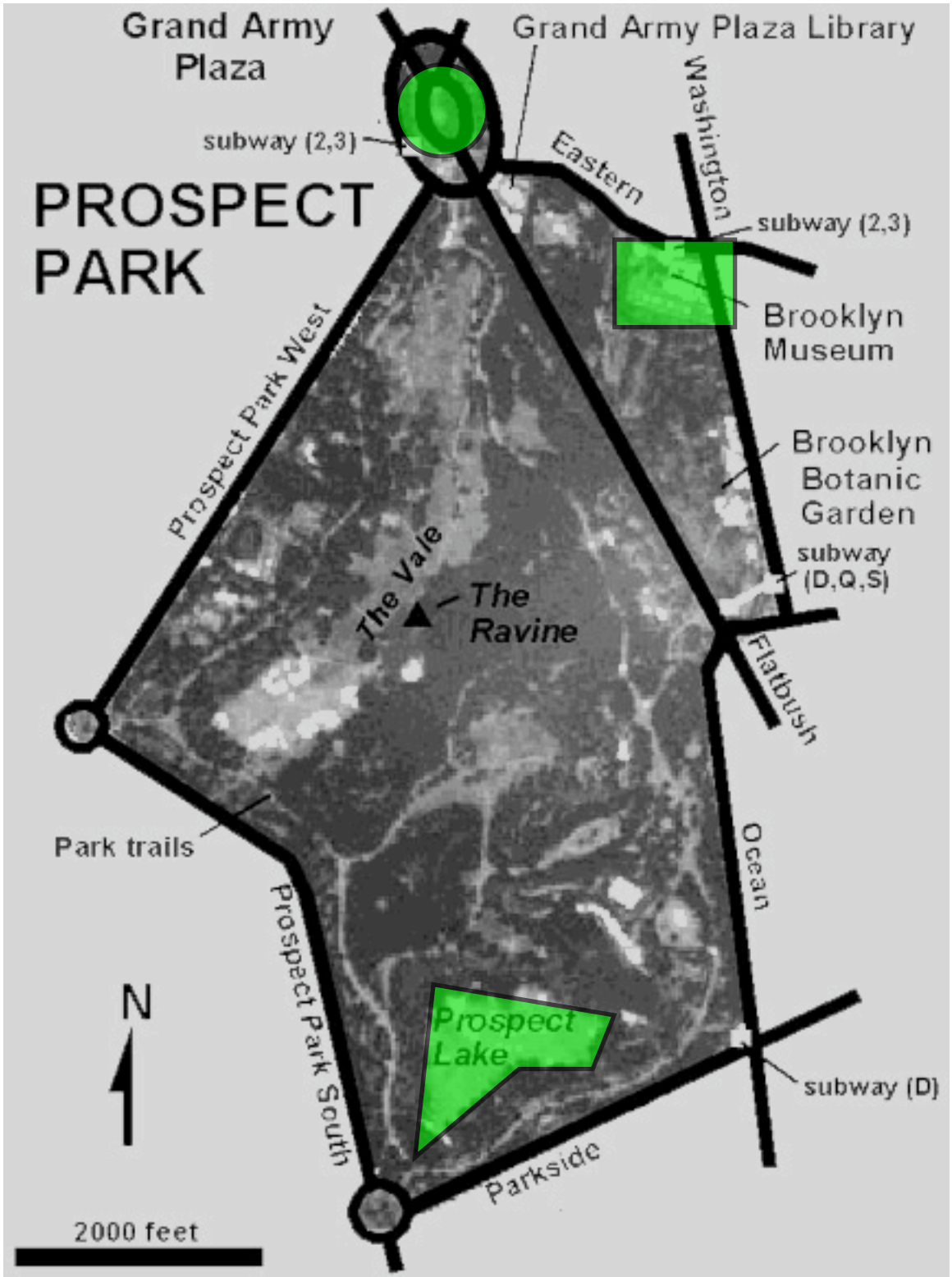


Origin is at the upper lefthand corner pixel.

Specify areas that are "*clickable*" on the map.

- `circle`
- `rect`
- `poly`

Use Ruler (or image program to find pixel coordinates)

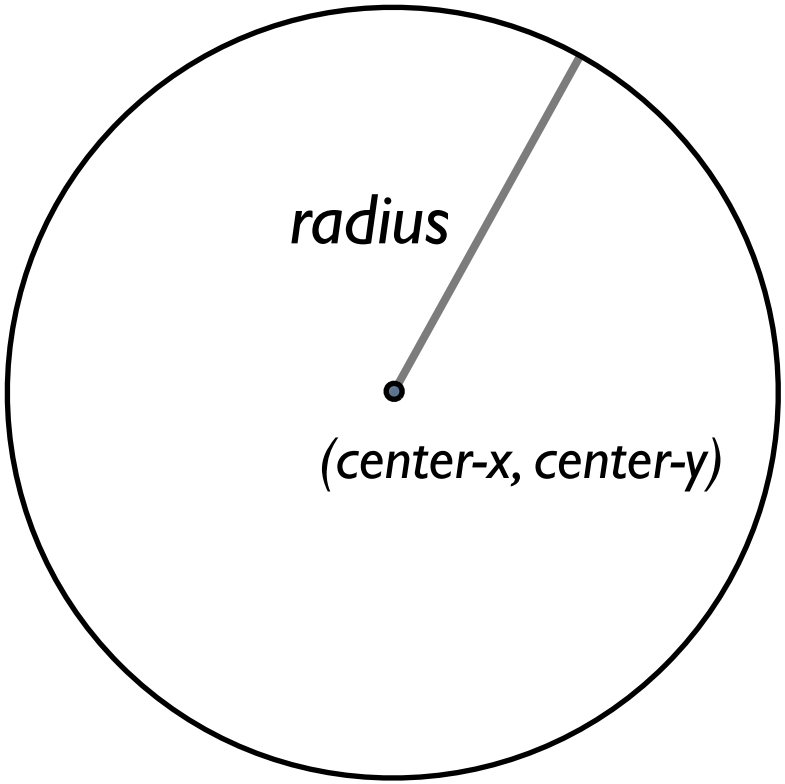


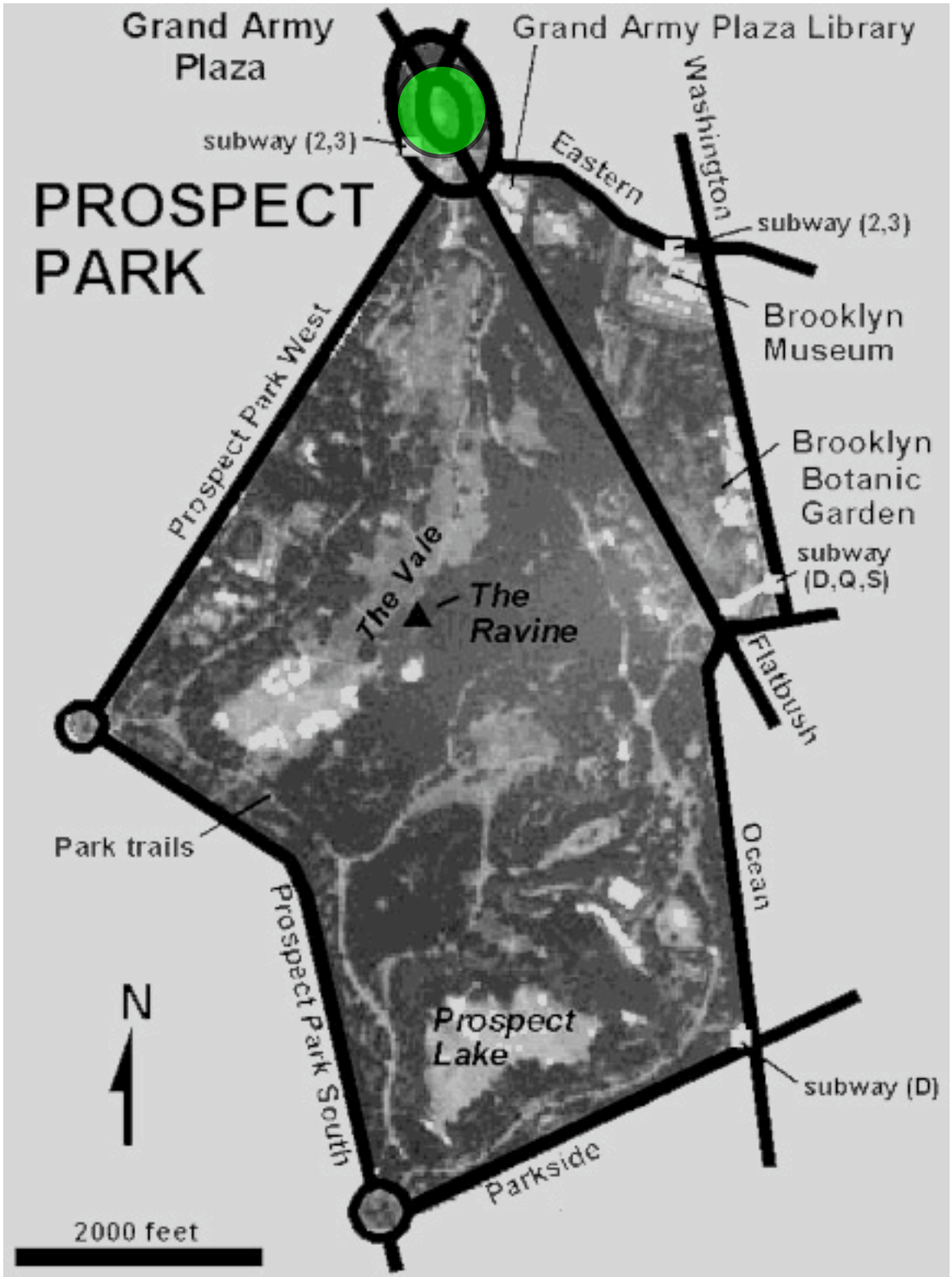
-circle:

center-x, center-y, radius

- example:

200, 50, 12



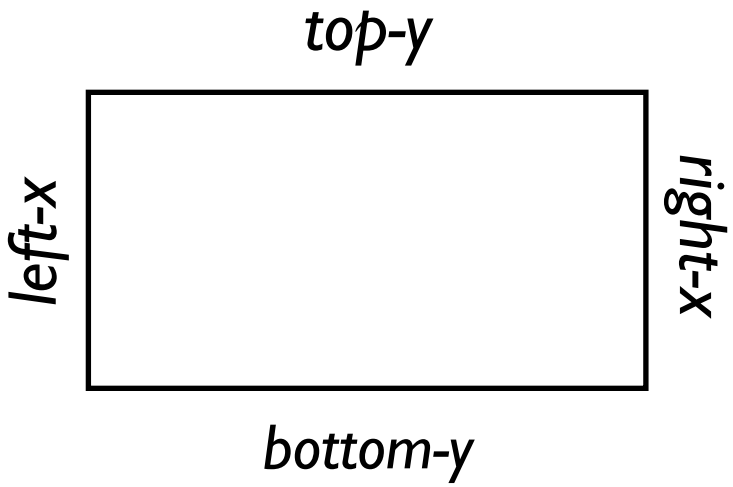


- **rect:**

left-x, top-y, right-x, bottom-y

- **example:**

288, 110, 333, 144

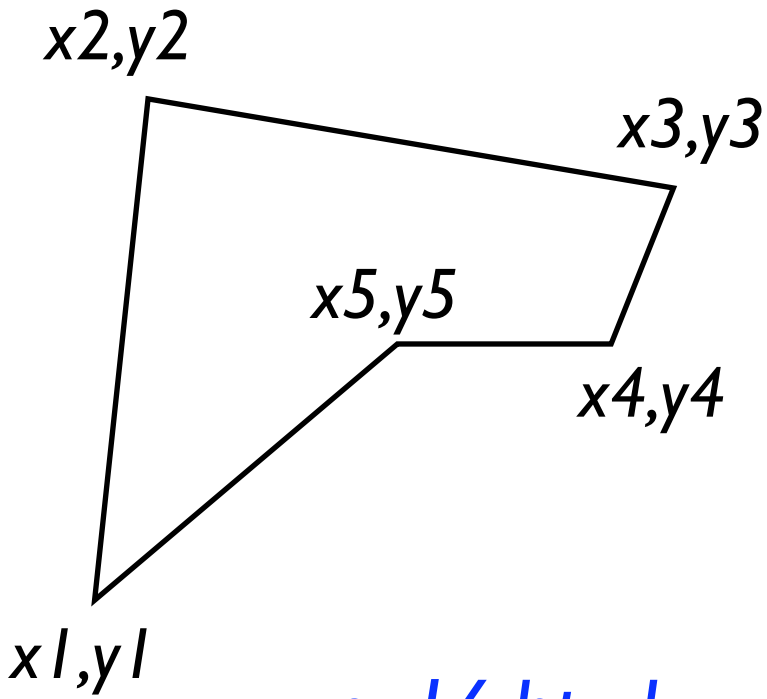


- **poly:**

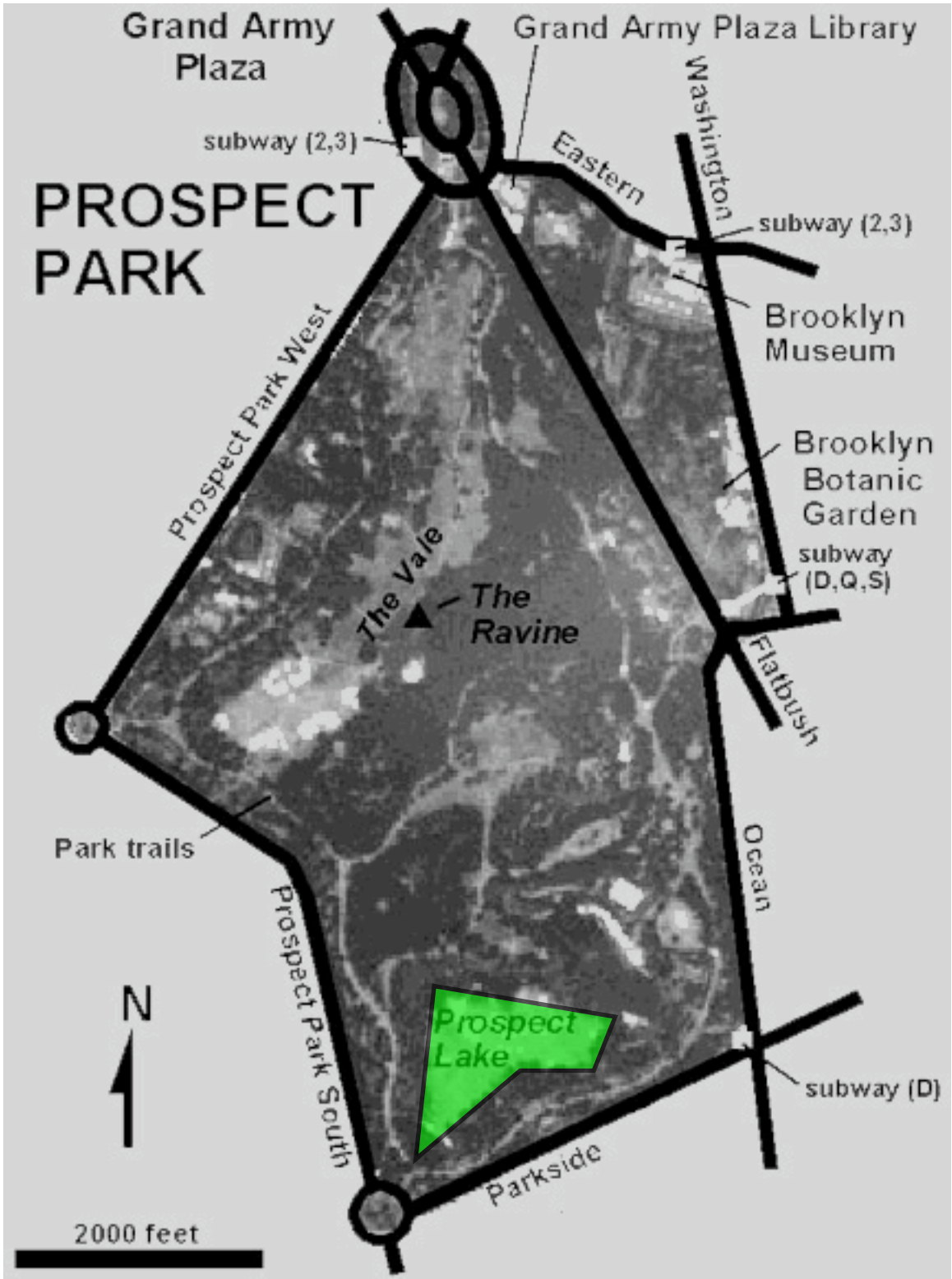
x1,y1,x2,y2, ... xn,yn

- **example:**

```
196,516,206,453,276,465,
266,486,233,487
```



ex16.html



Forms And Javascript

- overview

  – HTML forms provide a way to create dynamic web pages—i.e., the user has control over what appears

  – forms are linked with some type of program that runs either on the *server-side* or the *client-side*

  – we will learn a little bit of Javascript, which runs on the client side

- form tag (appears in the body of an HTML file)

```
<form name="form1">
.
.
.
</form>
```

Input Tags and Javascript

- input tag:

– contains input components that allow the user to enter information

```
<input type="button"
   name="mybutton"
   value="click me"
   onclick="alert('hello')" />
<input type="text" name="myname" />
```

ex27.html

ex28.html

- javascript functions handle user input, when user clicks on a button

– in the example above, the javascript function **`alert()`** is "called", or "run" or "invoked"

JavaScript

```
<html>

<body>

<script language=javascript>

//says the language is javascript

  //prints Hello World!

  document.write("Hello World!");


//stop JavaScript Code

</script>

</body>

</html>
```

Program Translation

- translates ***assembly*** or ***high-level languages*** into ***binary machine language***

- two methods:

  - ***interpretation***:

    reads and translates statements one at a time; doesn't optimize across an entire program; doesn't store executable statements— just runs them; error checking only happens at "*run time*", run time can be slow, but there's no "*compile time*"

  - ***compilation***:

    reads and translates entire program, and stores result as an executable file; can optimize; can perform "*compile time*" error checking; run-time is fast, but there is "*compile time*"

Compile-time vs. Compile Time

– *compile-time (noun)*:

the process of compiling a program from an assembly or high-level language into binary machine language and storing it on the computer's hard disk

– *compile time (adj noun)*:

the amount of time it takes a compiler to translate (or "compile") a program

Run-time vs. Run Time

– *run-time (noun):*

the process of executing a compiled, stored program

– *run time (adj noun):*

the amount of time it takes a program to run this is where **Big-Oh** comes in

Errors

– *errors:* can be found at compile-time and at run-time

– *error checking* : is done at compile-time

Data Representation

- bits
  - a bit is the smallest unit of memory
  - bit = **b**inary dig**it**
  - a bit is a switch inside the computer; the setting (or value) of each switch is either ON (= 1) or OFF (=0)
  - all data in a computer is represented by bit patterns, i.e., sequences of 0's and 1's
  - all numbers can be represented by 0's and 1's in ***base 2***
  - hence the term *binary* computer!

- **bytes**

– a *byte* is a sequence of 8 bits

– thus there are $2^8 = 256$ possible values that can be represented by one byte

– values range from 0 to $2^8 - 1$ (= 256 – 1 = 255)

– where 0 = 00000000 and 255 = 11111111

- ***trivia***

nybble = a sequence of 4 bits

base 2

– in base 2, only the digits 0 and 1 are used

– just like base 10, each digit, from the right to the left, indicates how many of each base raised to a power are contained in the number that is represented...

* note that digits are counted from right to left, starting with 0 to convert a byte to base 10, multiply each digit in the byte by the value in the table below, then add them all together

digit	7	6	5	4	3	2	1	0
power	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
value	128	64	32	16	8	4	2	1

Example of Binary to Decimal Conversion

Convert **00001011**, look up each digit in the table, and you add values of the places which contain a 1:

digit	7	6	5	4	3	2	1	0
power	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
value	128	64	32	16	8	4	2	1
x byte	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**
=	0 X 128	0 X 64	0 X 32	0 X 16	1 X 8	0 X 4	1 X 2	1 X 1
=	0	+ 0	+ 0	+ 0	+ 8	+ 0	+ 2	+ 1
=	11 (in base 10)							

Grouping Digits

– **base 8**, or *octal*—

since $2^3 = 8$, it is often convenient to compress 3-digit binary values as base 8, or octal values

– **base 16,** or *hexadecimal*—

since $2^4 = 16$, it is often convenient to compress 4-digit binary values as base 16, or hexadecimal values (also known as hex - remember the color numbers in HTML?)

Memorize values from 0 to 15

base 10	base 2	base 8	base 16
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Storing Numbers on a Computer

- **bytes** express values between 0 and 255 (called 8-bit)

- 32-bit **integers** store larger numbers btw. 0 and 4,294,967,295

- **negative numbers** - uses *2's complement notation*

  - the **sign bit**: left-most bit of the number indicates a negative sign or not

  ex: 11111111 = -1 in 2's complement

- **real numbers** - *floating point notation* - (floats) - numbers with a decimal point

  - store the whole and fractional part of the number seperately

  - uses scientific notation (exponents)

Storing Letters and Text

- letters, or ***characters***, are stored as numbers, but are encoded so that for each character on the keyboard (or displayed on the screen) there is a (positive) number that represents that character
- the software reading the value has to know that it should be interpreted as a character rather than a number
- the standard encoding is called **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)
  - standard ASCII encodes 128 characters
  - extended ASCII encodes 128 more, to total 256 characters

What's wrong with ASCII?

Unicode and Numbers

Unicode uses 2 bytes and encodes 2^{16}= 65536 characters (!) in many languages

"Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language."

(from ***http://www.unicode.org***)

– **NOTE** that the digits 0, 1, 2, ..., 9 can be stored as characters and have entries in the **ASCII** and **Unicode** tables

	00	10	20	30	40	50	60	70
0		ऐ	ठ	र	ी	ॐ	ॠ	॰
1	ँ	ऑ	ड	ऱ	ु	॑	ॡ	Res
2	ं	ऒ	ढ	ल	ू	॒	ॢ	Res
3	ः	ओ	ण	ळ	ृ	॓	ॣ	Res
4	ऄ	औ	त	ऴ	ॄ	॔	।	Res
5	अ	क	थ	व	ॅ	Res	॥	Res

File Storage

- files can be stored on a computer as either **plain text** or **binary**
- here is the simplest way to tell which a file's type is:
  - on Windows:

    go to the DOS prompt. enter:

    dos-prompt: **`type <filename>`**

    (where you substitute `<filename>` with the name of the file you want to check)

  - on Mac or UNIX:

    go to the terminal prompt. enter:

    unix-prompt: **`more <filename>`**

    (where you substitute **`<filename>`** with the name of the file you want to check)

  - on either operating system, text files will be displayed as letters and numbers that you can read and should look like what you expect; *binary files will look like garbage characters and might mess up your terminal window (in which case, just type* ***`reset`*** *and it will fix itself)*

- HTML files are plain text files
- most image files are binary files
- some files are stored as plain text, but their content is encoded so that you need special software to read the files
- usually, plain text files take up less space than binary files
- Exercise—

    1. Go into Word and create a new document. put the text "hello world" into the document and save it (as a word document).

    2. Then go into NotePad (or TextEdit on the Mac) and create a new document.

    3. put the same text ("hello world") into the document and save it (as plain text).

    4. Now look at both files in the explorer (or finder on the Mac) and compare their file sizes. ***which is larger? are you surprised?***

File Sizes

- file sizes are typically quoted in bytes, kilobytes, megabytes or gigabytes
- here are some handy conversions: (watch your BITS and BYTES...)

1 byte (B)	8 bits
1 kilobyte (KB)	1024 bytes = 2^{10} bytes
1 Mega Byte (MB)	1024 KB = 1024 X 1024 bytes = 2^{10} x 2^{10} bytes = 2^{20} bytes
1 Gigabyte (GB)	1024 MB = 1024 X 1024 X 1024 bytes = 2^{10} x 2^{10} x 2^{10} bytes = 2^{20} bytes

Bandwidth

- **bandwidth** = speed of data transmission

- data is transmitted at speeds that are measured in terms of **kilobits per second** (*kbits/s*)

(1 kilobit = 1000 bits = 10^3 bits $\approx$

1024 bits = 2^{10} bits = 2^7 bytes...)

- the time it takes to **download** a file (copy it from one computer to another) depends on the size of the file, the speed of the source computer (e.g., a server), the speed of the network and the speed of the destination computer (e.g., your laptop or desktop)

Different Ways of Connecting to the Internet

– **dial-up (modem)**

typically 28.8K (kilobits/sec) or 28, 000 bits per second or 56K (56,000 bps)

– **DSL = "Digital Subscriber Line"**

part of ISDN (Integrated Services Digital Network); allows data transmission over regular telephone lines

typically ranges from 128 K (kilobits/sec) to 24,000 K (kilobits/sec)

– **cable modem**

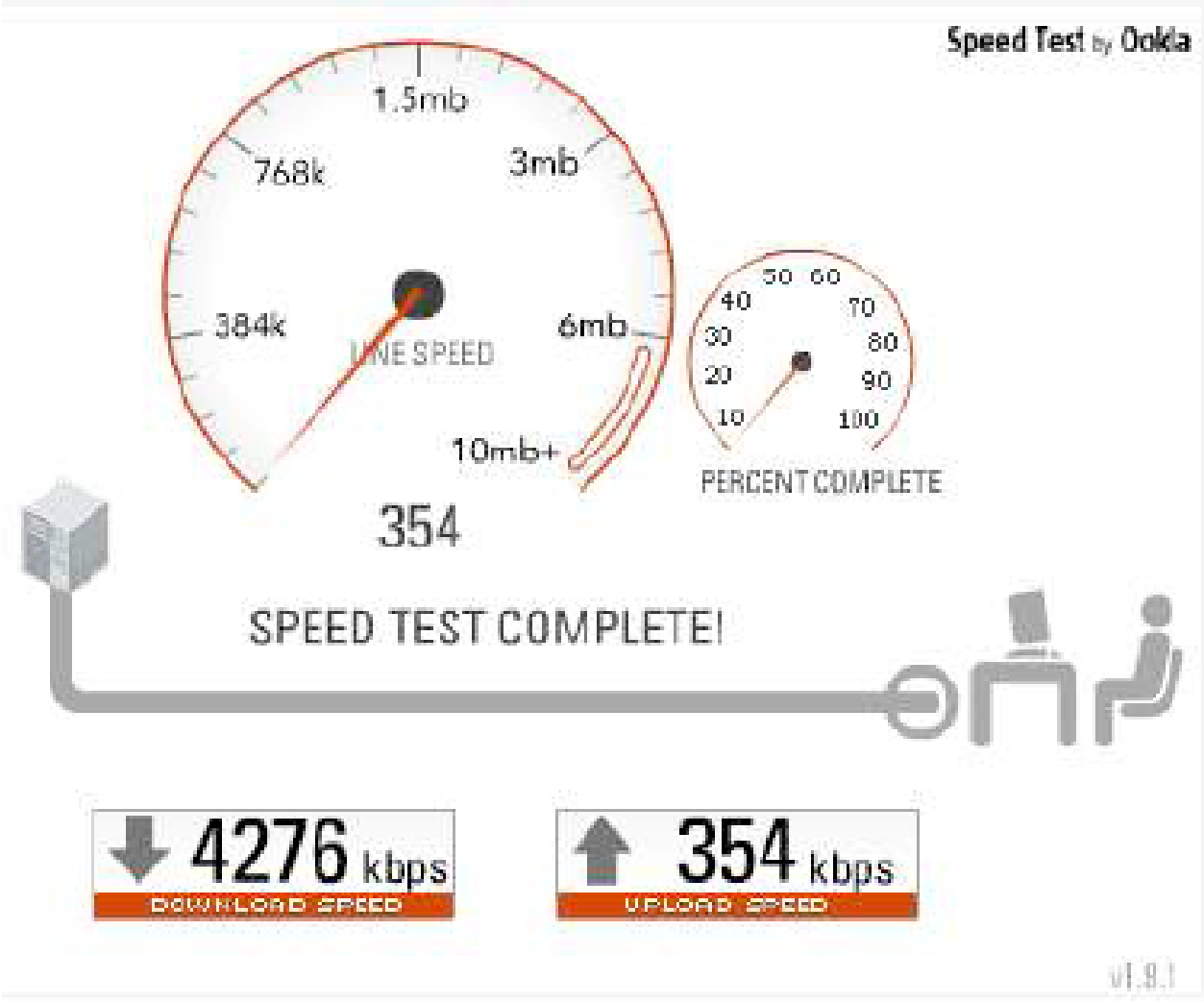
carries data transmissions over digital cable television lines

typically ranges from 384 K (kilobits/sec) to 30 M (megabits/sec) for a fast business line

Exercise at Home

- take the speakeasy speed test:

`http://www.speakeasy.net/speedtest/`



TO DO

READ: Chapter 12 and Chapter 14 (*concepts only*)

READ: Hackers and Painters (*Case Studies On-line.*)

There are only 10 different kinds of people in the world: those who know binary and those who don't.

- Anonymous

Exercise: Convert 13 base 8 to base 10

$$13_{base8} = 13_{8}$$

$$= (1 \times 8^{1}) + (3 \times 8^{0})$$

$$= (1 \times 8) + (3 \times 1)$$

$$= 8 + 3$$

$$= 11$$