Review Web Search

Functions & Functional Abstraction & More Functions
1. Gather information.
2. Keep copies.
3. Build an index.
4. Understand the query.
5. Determine the relevance of each possible result to the query.
6. Determine the ranking of the relevant results.
7. Present the results.
Clickers: It Matters How It Works

“The search tools that help us find needles in the digital haystack have become the lenses through which we view the digital landscape. Businesses and governments use them to distort our picture of reality.”

A. TRUE
B. FALSE
Clickers

- Google responds efficiently to queries by going out and searching the web in real time.

A. TRUE.
B. FALSE
Which steps happen in the background?

A. Gather information & Keep copies.
B. Gather information & Keep copies & Build an index.
C. Understand the query.
D. Determine the relevance of each possible result to the query & Determine the ranking of the relevant results.
How does it find the pages to index?

A. Every website has to register with Google to get indexed
B. Every website has to pay Google to get indexed
C. Google knows which pages to index because of your digital footprint
D. Google has a list of ‘trusted pages’ and it just follows the links from them
Google indexes what percent of the pages in the world?

A. 100% It indexes everything.
B. About 50% on average but some days it’s 100% and some days it’s 30%
C. Less than 5%
D. Best estimates put it at 30% because of all the bots that keep crawlers out.
Clickers: What goes into the index?

A. Every word on the page
B. The words that other sites use when they point to (link to) this page.
C. Only the keywords on the page.
D. Google has a list of special keywords that all pages get indexed by.
How often does a page get visited?

A. Every page that gets indexed is visited every day.
B. It depends on the page, pages like whitehouse.gov get visited daily others rarely
C. Google decides by keeping track of how often pages change.
D. B & C
E. None of the above.
Web pages typically represent the present but they could be a few days out of date.

A. TRUE
B. FALSE
Web pages typically represent the present but they could be a few days out of date.

University of Sheffield

HUMAINE researchers at Sheffield University Computer Science are drawn from the Natural Language Processing (NLP), Speech & Hearing (SpandH) and Computer Graphics research groups.

The NLP/SpandH conjugation is the largest in the UK, and specialises in dialogue interfaces as well as a range of topics, such as information extraction, information management, ontology induction, and speech recognition areas, in which it has achieved some of the best scores internationally in US DARPA competitions. Its expressive dialogue work dates back to 1997, when members of the NLP group designed the CONVERSE system that entered the Loebner Competition in New York for the most plausible computer dialogue partner of the year and won. They then began to organise (with EC funding) a series of international workshops on dialogue at Bellagio that brought together industrial and academic work on machine dialogue. The group also participates in a range of EPSRC projects (the most prestigious form of research funding in the UK) including AKT, a five-University six-year interdisciplinary research consortium. SpandH are scientific coordinators of AMI, a 6FP Integrated project, on understanding multispeaker meetings.

The Computer Graphics Research Group (see Computer Graphics) is long established, having amongst its members Alan Watt, the author of many famous 3D graphics books. With the acquisition of the new immersive Virtual Reality lab and the facilities at the North Campus has taken a new direction focusing on the creation of Virtual Environments, including the creation of virtual animated characters thier integration with computer dialogue systems.

MAIN MEMBERS

Yorick Wilks is head of the NLP group and the author of 5 books and hundreds of papers in the area. His Cambridge degrees were first in mathematics and philosophy, and then a PhD in computing and philosophy. He has been a researcher at Cambridge, Stanford, Edinburgh, Essex, New Mexico, Oxford and Sheffield Universities. He is a Fellow of both the European and American Artificial Intelligence Societies, a Fellow of the EPSRC College of Computing and a member of the UK Computing Research Council. He designed the CONVERSE system, the EVDGEC belief and intention modelling system, and the COMPAIONS paradigm of an intelligent and affective agent, designed as a permanent but computer-based personality functioning as a companion to a particular person.

Daniela Romano is a lecturer of the Computer Graphics research group and has obtained a Ph.D. from the University of Leeds, UK, and an MSc in Computer Science University from the University of Bari, Italy. Her career in computer science started with six years in the software development for private sectors companies (including Microsoft S.p.A. and the Ford Motor Company) in Italy, Germany and the UK. Daniela's academic career started in 1996. Before joining the Multimedia Graphics Group at Sheffield, she worked for the Universities of Leeds, University College London and Salford. Daniela specialised in the creation of educational virtual environments for training or therapeutic purposes and in particular in the creation of empathic animated characters, displaying emotions and personality.

Marilyn Walker is Royal Society Wolfson Professor in Computer Science at the University of Sheffield. She has an M.S. in Computer Science from Stanford University, and an M.A. in Linguistics and a Ph.D. in Computer Science from the University of Pennsylvania. Her research focuses on spoken dialogue systems, especially on spoken language generation and the use of machine learning in dialogue. As a member of the Speech and Information Processing Lab at AT&T Bell Labs, she built the first dialogue system to use reinforcement learning to adapt the dialogue manager on the basis of interaction with human users, and developed the first trainable generation component for a spoken dialogue system as part of her work on the DARPA sponsored AT&T Communicator project. Her research on spoken language generation has examined the effect of social relationship and cognitive factors on language production.

Roger Moore took up a Chair in SpandH in August 2004 having been Chief Scientific Officer of 20/20 Speech Ltd. since 1999. Prior to that, he was Head of the UK Government’s Speech Research Unit (SRU) from 1985. He has a BA in Computer & Communications Engineering, and an MSc and PhD in Automatic Speech Recognition from Essex University. He has authored over 100 scientific publications in Speech Technology algorithms, applications and assessment. He is Editor of Computer Speech & Language, a member of the Editorial Board for Speech Communication, a Fellow of the UK Institute of Acoustics and a Visiting Professor in the Department of Phonetics and Linguistics at University College London. He was President of the ‘European/International Speech Communication Association’ (ISCA) from 1997 to 2001.
Web pages typically represent the present but they could be a few days out of date.
Clickers

- The federal government uses your tax dollars to guarantee that Google and other search engine providers like Microsoft (Bing) return the objectively best results of your query.

A. TRUE

B. FALSE
How does the index get used at search time by default? (without advanced search)

A. Every website registers with Google exactly which terms to index by and which combinations

B. Google uses “boolean” combinations. The index is made of single words. Google ANDS them together and finds which webpages (URLs) are in the intersection of all the terms

C. Google indexes the pages individually for each person using your digital footprint

D. All of the above.
Clickers

- Web search is free and democratic. Every web page has an equal chance of being indexed and coming out the top of the list.

A. TRUE  
B. FALSE
How does it rank the pages it finds?

- It uses what is called a “page rank” algorithm, that uses many different factors
- It depends only on who is willing to pay the most.
- The government tells it how to rank pages.
- None of the above.
The information that Page Rank uses includes:

- Keywords in heading or titles and keywords in the body text
- Information about whether the site is “trustworthy”
- Whether the links on this page are to relevant pages
- Whether the links to this page are relevant
- Age of the page
- Quality of the text (e.g. absence of misspellings)

It uses what is called a “page rank” algorithm, that uses many different factors.
Search engine possible funding models:

A. Users could pay a subscription fee
B. Web sites could pay for being indexed.
C. The government could pay using taxes the same way they pay for roads or police.
D. Advertisers could pay for having their ads featured in the side bar.
E. All of the above are possible.
Search engine current funding model:

A. Users pay a subscription fee
B. Web sites pay for being indexed.
C. The government pays using taxes the same way they pay for roads or police.
D. Advertisers pay for having their ads featured in the side bar.
E. B&D
What makes search attractive to advertisers?

A. Advertisements can be targeted more precisely using your digital footprint.
B. Advertisers only pay when you click on their ad.
C. Millions of people use search every day.
D. All of the above.
Functions and More Functions
Homework 10: Layering Functions

- Functions SUDOKU homework assigned next Tuesday for one week, start in advance if you can.

Sudoku GUI

The Sudoku board is a 9×9 array of cells. Alternating 3×3 cells are colored gray and strong lines separate all of the 3×3 cells, as follows. (More information at http://www.websudoku.com/.)
For loops (Repetition)

- Repeating commands is powerful:
  - Lightbot 2.0 used recursion, a function calling itself
  - Symbolic Lightbot prefixed a number, 2:Step
- Processing uses a `for` loop:

```java
void setup() {
  size(500,200);
  background(0);
  noStroke();
  smooth();
  fill(255);
  for (int i=0; i < 16; i++) {
    ellipse(100+25*i, 100, 15, 15);
  }
}
```
Using Functions

- Functions package up computation ... when do we use them? All the time.
- Write some simple code to achieve a goal ...

```java
void setup() {
  size(300, 300);
  background(102);
  noStroke();
  fill(255, 255, 0);
}

void draw() {
  rect(100, 100, 50, 50);
}
```
To put the rectangle in different places, we “parameterize” the position, that is, use input to the function to place the rectangle.
Once Created, Use It Everywhere

- Now we quit thinking of drawing a rectangle, but now think of placing a 50x50 rectangle.
We are going to make a Clock Timer.

- Draw digital timer elements
- Assemble elements into digits
- Light digit segments to create numbers
- Select number based on a digit
color dark, lite;

void setup() {
  size(400, 200);
  smooth();
  background(20);
  noStroke();
  frameRate(10);
  fill(dark);
  digit(50, 90);
  digit(140, 90);
}
Writing Functions

- Processing function definitions are typically listed after the standard blocks: setup(), draw(), mousePressed(), etc.

```java
void setup() {
    size(100, 100);
    background(0);
    noStroke();
}

void draw() {
    fill(255);
    hexa(20, 20);
}

void hexa(float xbase, float ybase) {
    rect(xbase, ybase+10, 20, 40);
    triangle(xbase, ybase+10, xbase+20, ybase+10, xbase+10, ybase);
    triangle(xbase, ybase+50, xbase+20, ybase+50, xbase+10, ybase+60);
}
```
Using Functions

- Once defined, functions can be called repeatedly ... it’s the point of writing them!

```c
void setup() {
    size(110, 100);
    background(0);
    noStroke();
}

void draw() {
    fill(255);
    hexa(20, 20);
    hexa(50, 20);
    hexa(80, 20);
}

void hexa(float xbase, float ybase) {
    rect(xbase, ybase+10, 20, 40);
    triangle(xbase, ybase+10, xbase+20, ybase+10, xbase+10, ybase);
    triangle(xbase, ybase+50, xbase+20, ybase+50, xbase+10, ybase+60);
}
```
```cpp
void setup() {
  size(110, 100);
  background(0);
  noStroke();
}

void draw() {
  fill(255);
  hexa(20, 20);
  hexa(50, 20);
  hexa(80, 20);
  // hexa(30, 20);
}
```
Clicker: if we didn’t have hexa

A. The draw function would be exactly like it is.

B. The code for hexa would be inside the draw function one time exactly as it appears here.

C. The code for hexa would be inside the draw function three times exactly as it appears here.

D. The code for hexa would be inside the draw function three times with different values given as arguments to rect and triangle.
Arguments Become Parameters

- Notice that if the DEFINITION has $n$ parameters, the CALL needs $n$ arguments
- The parameters and arguments correspond

Inside of the function, the parameter, e.g. xbase, is declared and initialized to the corresponding argument, e.g. 80. Then, the definition uses it, e.g.

```c
void draw( ) {
  fill(255);
  hexa(20, 40);
  hexa(50, 40);
  hexa(80, 40);
}
void hexa(float xbase, float ybase) {
  rect(xbase, ybase+10, 20, 40);
  triangle(xbase, ybase+10, xbase+20, ybase+10, xbase+10, ybase);
  triangle(xbase, ybase+50, xbase+20, ybase+50, xbase+10, ybase+60);
}
rect(80, 40+10, 20, 40)
```
More On Parameters …

- **Parameters**: Customize each function call to a specific situation – they are the input to the function
  - *Parameters* are the names of the input values used inside of the procedure body
  - *Arguments* are the values from outside to be used for each of the parameters
Example: Clock Timer. In sample codes.

- Draw digital timer elements
- Assemble elements into digits
- Light digit segments to create numbers
- Select number based on a digit
Functional Abstraction: Layers of Functions

- Review What We Did

- The computation is ONLY drawing triangles and rectangles, but we don’t think of it that way ... to us, it’s a timer
Count In Lights: a function for each number

- Light up the digit for each number
- Count_in_lights.pde

```cpp
void draw() {
    lite = color(255,185,0);
    dark = color(64, 48, 0);
    fill(dark);
    digit(50,20);
    fill(lite);
    two(150,20);
    //one(150,20);
}
```
Count in Lights. Layers of Functions

digit one two three four five
hexa rexa
triangle, rect, triangle
Define hexa( ) and rexa( )

- just_hexa_rexa.pde
- Parameterize the functions by a consistent position – upper left corner is good

```c
void draw() {
    fill(255);
    hexa(20, 40);
    rexa(30, 20);
}

void hexa(float xbase, float ybase) {
    rect(xbase, ybase+10, 20, 40);
    triangle(xbase, ybase+10, xbase+20, ybase+10, xbase+10, ybase);
    triangle(xbase, ybase+50, xbase+20, ybase+50, xbase+10, ybase+60);
}

void rexa(float xbase, float ybase) {
    triangle(xbase, ybase+10, xbase+10, ybase, xbase+10, ybase+20);
    rect(xbase+10, ybase, 40, 20);
    triangle(xbase+50, ybase, xbase+50, ybase+20, xbase+60, ybase+10);
}
```
Clicker: I have hexa( ) and rexa( ), now what?

A. I can make fun little shapes
B. It obvious how I can make digits given these two shapes
C. I have to define at least two more functions before I can make digits
D. I think I can make digits but I just can’t see how.
Use hexa, rexa to make a Digit

- hexa_rexa_makeadigit.pde

```plaintext
void hexa(float xbase, float ybase) {
    rect(xbase, ybase+10, 20, 40);
    triangle(xbase, ybase+10, xbase+20, ybase+10, xbase+10, ybase);
    triangle(xbase, ybase+50, xbase+20, ybase+50, xbase+10, ybase+60);
}

void rexa(float xbase, float ybase) {
    triangle(xbase, ybase, xbase+10, ybase-10, xbase+10, ybase+10);
    rect(xbase+10, ybase-10, 40, 20);
    triangle(xbase+50, ybase-10, xbase+50, ybase+10, xbase+60, ybase);
}

void digit(float xbase, float ybase) {
    hexa(xbase, ybase+10);    // left upper
    hexa(xbase, ybase+70);    // left lower
    rexa(xbase+10, ybase);    // top horizontal
    rexa(xbase+10, ybase+60); // mid horizontal
    rexa(xbase+10, ybase+120); // bot horizontal
    hexa(xbase+60, ybase+10); // right upper
    hexa(xbase+60, ybase+70); // right lower
}
```

2/20/13
Clicker: What does digit help you do?

A. You can re-use it over and over again
B. It tells you what the coordinates are for the various bits of the digit
C. If you pass the right arguments for xbase and ybase to digit you can make every digit
D. A & B
E. None of the above

void digit(float xbase, float ybase) {
    hexa(xbase, ybase+10);  //left upper
    hexa(xbase, ybase+70);  //left lower
    rexa(xbase+10, ybase);  //mid horizontal
    rexa(xbase+10, ybase+60);  //top horizontal
    rexa(xbase+10, ybase+120);  //bot horizontal
    hexa(xbase+60, ybase+10);  //right upper
    hexa(xbase+60, ybase+70);  //right lower
}
Let There Be Light (and Dark)

- Define the illumination of the digit
- hexa_rexa_makeadigit_addlight.pde
  - Must declare two color variables, initialize to proper colors, use them in fill, and then check them

```pde
color dark, lite;

void setup( ) {
  size(250, 180);
  background(0);
  stroke(0);
}

void draw( ) {
  lite = color(255, 185, 0);
  dark = color(64, 48, 0);

  fill(dark);
  digit(50, 20);
  fill(lite);
  digit(140, 20);
}
```
Clicker: What does the lighted digit help you do?

A. You can re-use it over and over again

B. It tells you what the coordinates are for the various bits of the digit

C. If you pass the right arguments for xbase and ybase to digit you can make every digit

D. A & B

E. None of the above

void digit(float xbase, float ybase) {
    hexa(xbase, ybase+10);       //left upper
    hexa(xbase, ybase+70);       //left lower
    rexa(xbase+10, ybase);       //mid horizontal
    rexa(xbase+10, ybase+60);    //top horizontal
    rexa(xbase+10, ybase+120);   //bot horizontal
    hexa(xbase+60, ybase+10);    //right upper
    hexa(xbase+60, ybase+70);    //right lower
}
Count In Lights: a function for each number

- Light up the digit for each number
- Count_in_lights.pde

```plaintext
void draw() {
    lite = color(255,185,0);
dark = color(64, 48, 0);
fill(dark);
digit(50,20);
    fill(lite);
    two(150,150);
    //one(15)
}

void one (float xbase, float ybase) {
    hexa(xbase+60, ybase);
    hexa(xbase+60, ybase+60);
}

void two (float xbase, float ybase) {
    rexa(xbase+10, ybase);  //top horizontal
    hexa(xbase+60, ybase);  // right upper
    rexa(xbase+10, ybase+60);  //mid horizontal
    rexa(xbase+10, ybase+120);  //bot horizontal
    hexa(xbase, ybase+60);  //left lower
}
```
void four (float xbase, float ybase) {
    hexa(xbase, ybase);    // left upper
    hexa(xbase+60, ybase);  // right upper
    rexa(xbase+10, ybase+10); // mid horizontal
    hexa(xbase+60, ybase+60); // right lower
}

A. Perfect just as it is
B. Has a bug for right upper
C. Has a bug for mid horizontal
D. Actually makes a five
void digit(float xbase, float ybase) {
    //hexa(xbase, ybase+10);  //left upper
    //hexa(xbase, ybase+70);  //left lower
    leftupper (xbase,ybase);
    leftlower (xbase,ybase);
    //hexa(xbase+10, ybase);  //top horizontal
    tophoriz (xbase,ybase);    //top horizontal
    rexaxbase+10, ybase+60);  //mid horizontal
    rexaxbase+10, ybase+120);  //bot horizontal
    hexa(xbase+60, ybase+10);  //right upper
    hexa(xbase+60, ybase+70);  //right lower
}

void leftupper (float xbase, float ybase) {
    hexa(xbase, ybase+10);  //left upper
}

void leftlower (float xbase, float ybase) {
}
Clicker: why would you want Count_in_lights_three_level?

A. I wouldn’t want it, it just makes the code even more confusing

B. Adding numbers to ybase and xbase for rexa and hexa can cause silly errors.

C. It makes the code more self documenting.

D. B & C
Example: Clock Timer. In sample codes.

- Draw digital timer elements
- Assemble elements into digits
- Light digit segments to create numbers
- Select number based on a digit
The computation is ONLY drawing triangles and rectangles, but we don’t think of it that way ... to us, it’s a timer
Select A Number To Display

- Given an integer, display it in lights

```c
void sel(int n, float xbase, float ybase) {
  fill(lite);
  if (n == 0) {
    zero(xbase, ybase);
  }
  if (n==1) {
    one(xbase, ybase);
  }
  if (n==2) {
    two(xbase, ybase);
  }
  if (n==3) {
    three(xbase, ybase);
  }
  if (n==4) {
    four(xbase, ybase);
  }
  if (n==5) {
    five(xbase, ybase);
  }
  if (n==6) {
    six(xbase, ybase);
  }
}```
Create a 3 Digit Display

```c
void three_digit(int n, float xbase, float ybase) {
    fill(dark);
    digit(50,90);
    digit(140, 90);
    digit(260, 90);
    fill(lite);
    rect(xbase+185, ybase+125, 15, 15);
    sel(((n/100)%10, xbase, ybase);
    sel(((n/10)%10, xbase+90, ybase);
    sel(n%10, xbase+210, ybase);
}
```

Here’s The Action
Count up At The Frame Rate

color dark, lite;
int i;

void setup( ) {
    size(400, 300);
    background(0);
    noStroke();
    frameRate(10);
}

void draw( ) {
    lite = color(255, 185, 0);
    dark = color(64, 48, 0);
    i = i + 1;
    three_digit(i, 50, 90);
}
What is time doing?

A. Nothing. It has a bug in it.
B. It is counting to infinity.
C. It is counting to 999 then starting at zero again
D. It counts up to 1000 then starts at 1.
What is time doing?

```c
void draw() {
lite = color(255, 185, 0);
dark = color(64, 48, 0);
if (stop == 1) {
    fill(dark);
digit(50, 90);
digit(140, 90);
digit(260, 90);
    fill(lite);
rect(235, 155, 15, 15);
time=(time+1)%1000;
    sel(time%10, 260, 90);
    sel((time/10)%10, 140, 90);
sel(time/100, 50, 90);
}
```
Functional Abstraction: Layers of Functions

three_digit

sel

digit one two three four five six seven eight nine zero

leftupper leftlower tophoriz midhoriz bothoriz rightupper rightlower

hexa rexa

triangle, rect, triangle
Homework 10: Functions

- void cell(int x, int y, int s, color tinto)
- void triple(int x, int y, int s, color tinto)
- Also functions for
  - block()
  - row()
  - cellarray()