



# COMP2720: Automating Tools for New Media

Introduction: Why study  
computer science at all?!?



# Story and objectives

- What is computer science about?
- What computers really understand
- Media manipulation: Why digitise media?
  - How can it possibly work?
- It's about communications and process
- Objectives: Learn what computer science is all about, how a computer works, and about encodings



# What's computation good for

- Computer science is the study of “*recipes*”
- Computer scientists study...
  - How the recipes are written (algorithms, software engineering)
  - The units used in the recipes (data structures, databases)
  - What can recipes be written for (systems, intelligent systems, theory)
  - How well the recipes work (complexity, human-computer interfaces)

# Specialised recipes

- Some people specialise in Thai food or barbeques
- Computer scientists can also specialise on special kinds of recipes
  - Recipes that create pictures, sounds, movies, animations (graphics, computer music)
- Still others look at *emergent properties* of computer “recipes”
  - What happens when lots of recipes talk to one another (networking, non-linear systems, complex systems)



# Key concept: The COMPUTER does the recipe!

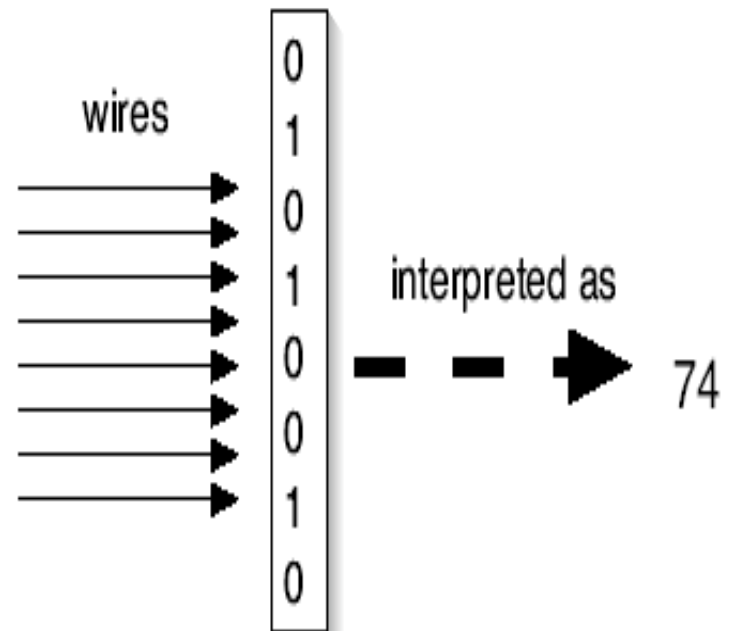
- Make it as hard, tedious, complex as you want!
- Process 100,000 credit card transaction per hour?  
Can and must do!
- Crank through a million genomes? Was a real feat!
- Find one person in a 10,000 campus? Easy!
- Process a million dots on the screen or a billion sound samples?
  - That's *media* computation

# What computers understand

- It's not really multimedia at all!
  - It's unimedia (*Nicholas Negroponte*, Media Lab, MIT)
  - Everything is 0's and 1's (0/1 = **bit** = binary digit)  
(this may change if Quantum Computer will be built; the **bit** will be replaced by **qubit**, which is more like a two dimensional sphere)
- Computers are not that smart:
  - The only data they understand is 0's and 1's
  - They can only do the most simple things with those 0's and 1's
    - Move this value here, read a value from one storage location, write a new value to another storage location
    - Add, multiply, subtract, divide these values
    - Compare these values, and if one is less than the other, go follow this step rather than that one.

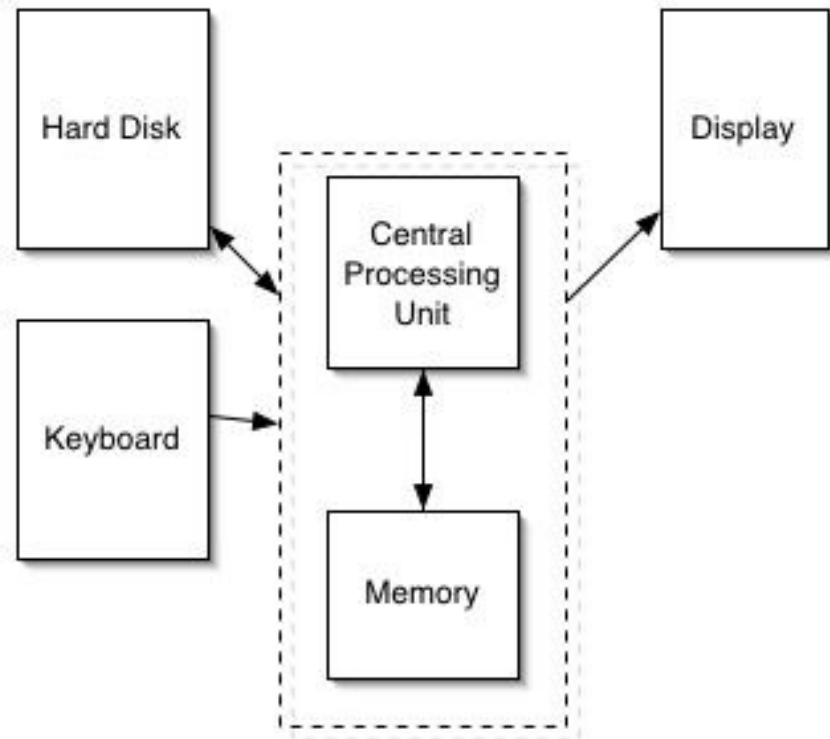
# Key concept: Encodings

- But we can interpret these numbers many ways as we need and can
  - We can encode information in those numbers
- Even the notion that the computer understands numbers is an interpretation
  - We encode the voltages on wires as 0's and 1's, eight of these (bits) define a **byte**
  - Which we can, in turn, interpret as a decimal number



# How a computer works

- The part that does the adding and comparing is the **Central Processing Unit** (CPU).
- The CPU talks to the memory
  - Think of it as a sequence of millions of mailboxes, each one byte in size, each of which has a numeric address
- The hard disk provides 10 times or more storage than memory (128 billion bytes versus 512 million bytes), but is millions of times slower
- The display is the monitor or LCD (or whatever)



# Layered computer architecture (P. Denning)

- |    |                            |            |
|----|----------------------------|------------|
| 15 | Workflow process           | $10^5$     |
| 14 | User interface             | $10^3$     |
| 13 | User virtual machine       | $10^1$     |
| 12 | Directories                | $10^{-1}$  |
| 11 | I/O streams                | $10^{-2}$  |
| 10 | Devices                    | $10^{-2}$  |
| 9  | Files                      | $10^{-2}$  |
| 8  | Interprocess communication | $10^{-2}$  |
| 7  | Virtual memory             | $10^{-2}$  |
| 6  | Local secondary storage    | $10^{-3}$  |
| 5  | Processes and semaphores   | $10^{-4}$  |
| 4  | Interrupts                 | $10^{-5}$  |
| 3  | Procedures                 | $10^{-6}$  |
| 2  | Instruction set            | $10^{-8}$  |
| 1  | Random access memory (RAM) | $10^{-8}$  |
| 0  | Hardware electronics       | $10^{-12}$ |

# Layer the encodings as deep as you want

- One encoding, ASCII, defines an “A” as 65
  - If there’s a byte with a value 65 in it, and we decide that it’s a character or string, POOF! It’s an “A”!
  - ASCII = American Standard Code for Information Interchange
- We can string lots of these numbers together to make usable text
  - “65, 172, 165, 184, 165, 169” is “Alexei”
  - “60, 97, 32, 104, 114, 101, 102, 61” is “<a href=“ (HTML)

# What do we mean by layered encodings?

- A number is just a number is just a number
- If you have to treat it as a letter, there's a piece of software that does it
  - For example, that associates 65 with the graphical representation for "A"
- If you have to treat it as part of an HTML document, there's a piece of software that does it
  - That understands that "<A HREF=" is the beginning of a link
- The part that knows HTML communicates with the part that knows that 65 is an "A"

# So in fact -- Multimedia is unimedia

- But that same byte with a 65 in it might be interpreted as...
  - A very small piece of sound (e.g., 1/44100-th of a second)
  - The amount of redness in a single dot in a larger picture
  - The amount of redness in a single dot in a larger picture which is a single frame in a full-length motion picture



# Software (recipes) defines and manipulates encodings

- Computer programs manage all these layers
  - How do you decide what a number should mean, and how you should organise your numbers to represent all the data you want?
  - That's data structures
- If that sounds like a lot of data, it is
  - To represent all the dots on your screen probably takes more than 3,145,728 bytes
  - Each second of sound on a CD takes 44,100 bytes (**what format?**)

# All formidable things we can now do with computers is owing to Moore's Law

- The statement known as Moore's law is a **prediction**
- It's just happened to come true ( for 30 odd years!)
- Not an ordinary achievement in a high-tech field
- The high-tech prediction most often fail (ridiculously) soon after they are issued:
  - × "Radio has no future." -- Lord Kelvin, inventor of the Kelvin scale, 1897
  - × "The cinema is little more than a fad...", Charlie Chaplin, 1916
  - × "The world potential market for copying machines is 5,000 at most." -- IBM to the founders of Xerox, 1959
  - × "We will never make a 32-bit operating system." -- Bill Gates, 1983
  - × "Two years from now, spam will be solved." -- Bill Gates, 2004
  - × "There's no chance that the iPhone is going to get any significant market share.", Steve Ballmer, USA Today, 2007

(From the web magazine *TechRadar* June 16 2008)

# Yet Moore's prediction became a law (and Thank God for that)

- Gordon Moore, one of the Intel founders, made the claim that (essentially) computer power doubles for the same dollar every 18 months.
- This has held true for over 30 years.
- Go ahead! Make your computer do the same thing to everyone of 3 million dots on your screen. It doesn't care! And it won't take much time either!
- Stupidity and speed were characterized by the famous physicist Richard Feynman as follows:
  - ``... a computer is dumb as hell but it goes like mad!''  
(and owing to the Moore law, one can say that every 18 month it gets twice as madder, but not a bit smarter)



# Why digitise media?

- Digitising media is encoding media into numbers
  - Real media is analogue (continuous).
  - To digitise it, we break it into parts where we can't perceive the parts.
- By converting them, we can more easily manipulate them, store them, transmit them without error, etc.



# How can it work to digitise media?

- Why does it work that we can break media into pieces and we don't perceive the breaks?
- We can only do it because human perception is limited.
  - We don't see the dots in the pictures, or the gaps in the sounds.
- We can make this happen because we know about physics (science of the physical world) and psychophysics (psychology of how we perceive the physical world)

# Why should you need to study “recipes”?

- To understand better the recipe-way of thinking
  - It's influencing everything, from computational science to bioinformatics
  - Eventually, it's going to become part of everyone's notion of a liberal education
  - That's the process argument
  - BTW, to work with and manage computer scientists
- **AND...to communicate!**
  - Writers, marketers, producers, communicate through computation



# Computation for communication

- All media are going digital
- Digital media are manipulated with software
- You are limited in your communication by what your software allows
  - What if you want to say something that Microsoft or Adobe or Apple doesn't let you say?



# Programming is a communications skill

- If you want to say something that your tools don't allow, program it yourself
- If you want to understand what your tools can or cannot do, you need to understand what the programs are doing
- If you care about preparing media for the Web, for marketing, for print, for broadcast... then it's worth your while to understand what the media are and how they can be manipulated.
- Knowledge is power. Knowing how media work is powerful and freeing
- (Richard Hamming): "The goal of computation is not numbers, but understanding"



# We're not going to replace PhotoShop

- Nor *ProAudio Tools*, *ImageMagick* and the *GIMP*, or *Java* and *Visual Basic*
- But if you know what these things are doing, you have something that can help you learn new tools

# Knowing about programming is knowing about processes

- Alan Perlis

- One of the founders of computer science
- Argued in 1961 that Computer Science should be part of a liberal education: Everyone should learn to program.
  - Perhaps computing is more critical to a liberal education than Calculus
  - Calculus is about rates, and that's important to many.
  - Computer science is about process, and that's important to everyone.





# A recipe is a statement of process

- A recipe defines how something is done
  - In a programming language that defines how the recipe is written
- When you learn the recipe that implements a *PhotoShop* filter, you learn how *PhotoShop* does what it does.
- And that is powerful.

# Finally: Programming is about communicating processes

- A program is the most concise statement possible to communicate a process
  - That's why it's important to scientists and others who want to specify how to do something understandably in as few words as possible

## Yet another argument (due to Alan Kay):

- The ability to “read” a medium means you can access materials and tools created by others. The ability to “write” in a medium means you can generate materials and tools for others. You must have both to be literate. In print writing, the tools you generate are rhetorical; they demonstrate and convince. In computer writing, the tools you generate are processes; they simulate and decide

“Making processes that simulate and decide requires programming”.

# Python



- The programming language we will be using is called Python

We didn't invent Python—it was invented by researchers across the Internet under the leadership of Guido van Rossum (who is inventor of Python and benevolent dictator for life)

- Home page: <http://www.python.org>
- It's used by companies like *Google, Industrial Light & Magic, NASA*, and organisations like *ATO, NSW Health, ABS*, and many others
- The kind of Python we're using is called Jython
  - It's Java-based Python (the Python proper is based on C/C++)
  - Home page: <http://www.jython.org>
- We'll be using a specific tool to make Python programming easier, called JES (Jython Environment for Students).