CSC 221: Computer Programming I

Fall 2006

Computer basics and history

- hardware vs. software
- generations of computer technology
- evolution of programming
- why Java?
- exploring object concepts using Alice
  - class, object, field, method

hardware vs. software

basic terminology:

- hardware – the physical components of the computer

  e.g., processor (Pentium 4, Celeron, Athlon, PowerPC, Alpha)
  memory (RAM, cache, hard drive, floppy drive, flash stick)
  input/output devices (keyboard, mouse, monitor, speaker)

- software – programs that run on the hardware

  e.g., operating system (Windows XP, Mac OS X, Linux)
  applications (Word, Excel, Powerpoint, RealPlayer, IE, Mozilla)
  development tools (JDK, BlueJ, .NET, CodeWarrior)

The easiest way to tell the difference between hardware and software is to kick it. If it hurts your toe, it's hardware.

Carl Farrell

History of computing technology

DYK?

When were "modern" computers invented?
When were computers accessible/affordable to individuals?
When was the Internet born?
When was the Web invented?
How did Bill Gates get so rich?

the history of computers can be divided into generations, with each generation defined by a technological breakthrough

0. gears and relays
   ➔ 1. vacuum tubes
      ➔ 2. transistors
         ➔ 3. integrated circuits
            ➔ 4. very large scale integration
               ➔ 5. parallel processing & networking

Generation 0: Mechanical Computers (1642-1945)

1642 – Pascal built a mechanical calculating machine

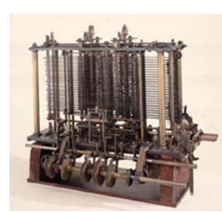
- mechanical gears, hand-crank, dials and knobs
- other similar machines followed

1805 – first programmable device, Jacquard loom

- wove tapestries with elaborate, programmable patterns
- pattern represented by metal punch-cards, fed into loom
- could mass-produce tapestries, reprogram with new cards

mid 1800's – Babbage designed "analytical engine"

- expanded upon mechanical calculators, but programmable via punch-cards
- described general layout of modern computers
- never functional, beyond technology of the day



Generation 0 (cont.)

1890 – Hollerith invented tabulating machine

- used for 1890 U.S. Census
- stored data on punch-cards, could sort and tabulate using electrical pins
- finished census in 6 weeks (vs. 7 years)
- Hollerith's company would become IBM

1930's – several engineers independently built "computers" using electromagnetic relays

- physical switch, open/close via electrical current

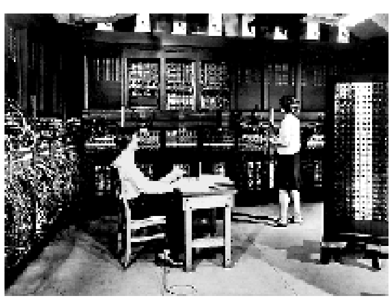
- Zuse (Nazi Germany) – destroyed in WWII
- Atanasoff (Iowa State) – built with grad student
- Stibitz (Bell Labs) – followed design of Babbage

5

Generation 1: Vacuum Tubes (1945-1954)

mid 1940's – vacuum tubes replaced relays

- glass tube w/ partial vacuum to speed electron flow
- faster than relays since no moving parts
- invented by de Forest in 1906



1940's – hybrid computers using vacuum tubes and relays were built

COLOSSUS (1943)

- built by British govt. (Alan Turing)
- used to decode Nazi communications

ENIAC (1946)

- built by Eckert & Mauchly at UPenn
- 18,000 vacuum tubes, 1,500 relays
- weighed 30 tons, consumed 140 kwatts

6

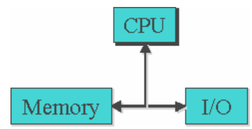
Generation 1 (cont.)

COLOSSUS and ENIAC were not general purpose computers

- could enter input using dials & knobs, paper tape
- but to perform a different computation, needed to reconfigure

von Neumann popularized the idea of a "stored program" computer

- store both data and programs in Memory
- Central Processing Unit (CPU) executes by loading program instructions from memory and executing them in sequence
- interact with the user via Input/Output devices



virtually all modern machines follow this *von Neumann Architecture*

programming was still difficult and tedious

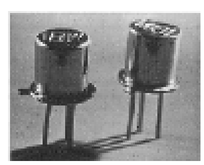
- each machine had its own *machine language*, 0's & 1's corresponding to the settings of physical components
- in 1950's, *assembly languages* replaced 0's & 1's with mnemonic names

7

Generation 2: Transistors (1954-1963)

mid 1950's – transistors began to replace tubes

- piece of silicon whose conductivity can be turned on and off using an electric current
- smaller, faster, more reliable, cheaper to mass produce
- invented by Bardeen, Brattain, & Shockley in 1948 (won 1956 Nobel Prize in physics)



computers became commercial as cost dropped

high-level languages were designed to make programming more natural

- FORTRAN (1957, Backus at IBM)
- LISP (1959, McCarthy at MIT)
- BASIC (1959, Kemeny at Dartmouth)
- COBOL (1960, Murray-Hopper at DOD)

the computer industry grew as businesses could buy
Eckert-Mauchly (1951), DEC (1957)
IBM became market force in 1960's

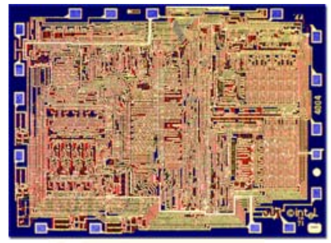
8

Generation 3: Integrated Circuits (1963-1973)

integrated circuit (IC)

- as transistor size decreased, could package many transistors with circuitry on silicon chip
- mass production further reduced prices

1971 – Intel marketed first *microprocessor*, the 4004, a chip with all the circuitry for a calculator



1960's saw the rise of Operating Systems

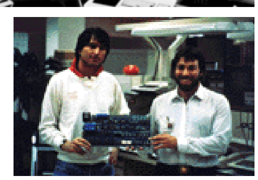
- an *operating system* is a collection of programs that manage peripheral devices and other resources
- allowed for *time-sharing*, where users share a computer by swapping jobs in and out
- as computers became affordable to small businesses, specialized programming languages were developed
  Pascal (1971, Wirth), C (1972, Ritche)

9

Generation 4: VLSI (1973-1985)

Very Large Scale Integration (VLSI)

- by mid 1970's, could fit hundreds of thousands of transistors w/ circuitry on a chip
- could mass produce powerful microprocessors and other useful IC's
- computers finally affordable to individuals



late 1970's saw rise of personal computing

- Gates & Allen founded Microsoft in 1975
  Gates wrote BASIC compiler for personal computer
  would grow into software giant, Gates richest in world
  http://evan.quuxuum.org/bgnw.html
- Wozniak and Jobs founded Apple in 1977
  went from garage to $120 million in sales by 1980
- IBM introduced PC in 1980
  Apple countered with Macintosh in 1984
- Stroustrup developed C++ in 1985
  object-oriented extension of C language

10

Generation 5: Parallelism & Networking (1985-????)

high-end machines (e.g. servers) can have multiple CPU's

- in 1997, highly parallel Deep Blue beat Kasparov in speed chess match

Year	Computers on the Internet	Web Servers on the Internet
2006	439,286,364	88.166,395
2004	285,139,107	52,131,889
2002	147,344,723	37,235,470
2000	93,047,785	18,169,498
1998	36,739,000	4,279,000
1996	12,881,000	300,000
1994	3,212,000	3,000
1992	992,000	50
1990	313,000	
1988	56,000	
1986	5,089	
1984	1,024	
1982	235	
1969	4	

most computers today are networked

- Internet born in 1969, connected 4 computers (UCLA, UCSB, SRI, & Utah)
  - mainly used by govt. & universities until late 80's/early 90's
- Web invented by Berners-Lee at CERN in 1989
  - designed to allow physics researchers to share data and documents
  - not popular until 1993 when Andreessen developed graphical browser (Mosaic)
  - Andreessen would go on to found Netscape, and Internet Explorer soon followed

stats from Internet Software Consortium & NetCraft

11

Evolution of programming: machine language

late 40's / early 50's: programmers coded directly in machine language

- each machine had its own set of instructions (sequences of 0's & 1's) corresponding to its underlying hardware

→ extremely tedious, error-prone

01111111010001010100110001000110000000010000000100000000100000000000000000
000000000000000000000000000000000000000000000000000000000000000000100000
000000000100000000000000000000000000000100000000000000000000000000000000
000000000000000000000000000000000000000000000000000000101000010000000000
000000000000000000000000000000001101000000000000000000000000000000000000
000000010100000000000000001000000000000000000000100000000001011100111001101101
000011100110111010001110010011101000110000101100010000000000001011100111010
001100101011110000111010000000000001011100111001001101111011001000110000010
111010001100001000000000010111001110011011110010110110101110100011000010 11
000100000000000101110011100110111010001110010011101000110000101100010000000
000001011100111001001100101011011000110000100101110011101000110010101111100
001110100000000000010111001100011011011110110110101101101011001010110111 00
1110100000000000000000000000000000010011101111000111011111111001000000000
1001100000000000000000000001001000000010010011000000000000000001010100000
0000000000000000010010010000100101010000000000000001000000000000000000000
0000000000000000100000000000000000000000001010000000010000000000000000010001
00100000000100000000000000001000000001010100000000000000000000000010010010000
1001010100000000000000001000000000000000000000000000000000010000000000000
000000000001011000000001000000000000000010000100000000000000000000000100000000
100000000000000000000000100000011100011111100000000001000100000001111010000
0000000000000000000000000000000000000000000000010010000110010101101100011
0110001101111011101110110111101110010011011000110010000100001000000000000
00000000000000000000000000000000000000000000001000000000000000000000000
0000000000000000000000000000000000000000000100000000000111111111100010
00000000000000000000000000001000000000000000000000000000000000000000000
0000000000000000000000001000000000011111111111000100000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000011000000000000000000000110000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
001000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000011000000000000000000000001000000000000000
000000000001101000000000000000000000000000000000000000000000000000000000
000000000100000000000000000000000000000000000000000000000000000001000100000
000000000000000000000000000000000000000000000000000000000000000001000000000
000000000000000000000000000000000000000000001000110000000000000000000000000
000000000000000000000000000000000000000000001000000000000000000000000000000
000000000000000000000000101100000000000000000000000000000000000000000000000
000000000000000000000000100000000000000000000000000000000000000000000000000
000110100100000000000000000000000000000000000000000000000000000000001001000
000100100000000000000000000000001000000000000000000000000000001101110000000000
000000000000000000000000000000000000000000000000000000000100000000000000000
00000000000000000000011010000110010101101100011011000110111100101110011000
1101110000011100000000000000011001110110001101100011001100100101111101100011
0110111101101101011100000110100101101100011001010110010000101110000000000001
0111110101000101011111011100010111010001101111011001000000000000101111110101
11110110110001110011010111110101111100110111011011110111001101110100011100
10011001010110000101101101010100000100011001010010001101110110111101110011
01110100011100100110010101100001011011010101111101010010001101110110111101
11001101110100011100100110010101100001011011010000000000101111101011110110
11000111001101011111010111110011011101101111011100110111010001110010011001
01011000010110110101010000010000110110001100000000011001010110111001100100
01101100010111110101111101000110010100100011011101101111011100110111010001
110010011001010110000101101101000000000011011010110000101101001011011100000
00000110001101101111011101010111010000000000000000000000000000000000000000
000000000000000000000000000000000000

12

Evolution of programming: assembly language

mid 1950's: assembly languages replaced numeric codes with mnemonic names

- an *assembler* is a program that translates assembly code into machine code

  input: assembly language program
  output: machine language program

- still low-level & machine-specific, but easier to program

```
gcc2_compiled.:
        .global _Q_qtod
.section        ".rodata"
        .align 8
.LLC0:  .asciz  "Hello world!"
.section        ".text"
        .align 4
        .global main
        .type    main,#function
        .proc   04
main:   !#PROLOGUE# 0
        save %sp,-112,%sp
        !#PROLOGUE# 1
        sethi %hi(cout),%o1
        or %o1,%lo(cout),%o0
        sethi %hi(.LLC0),%o2
        or %o2,%lo(.LLC0),%o1
        call __ls__7ostreamPCc,0
        nop
        mov %o0,%l0
        mov %l0,%o0
        sethi %hi(endl__FR7ostream),%o2
        or %o2,%lo(endl__FR7ostream),%o1
        call
__ls__7ostreamPFR7ostream_R7ostream,0
        nop
        mov 0,%i0
        b .LL230
        nop
.LL230: ret
        restore
.LLfe1: .size    main,.LLfe1-main
        .ident  "GCC: (GNU) 2.7.2"
```


Evolution of programming: high-level language

late 1950's – present:
high-level languages allow the programmer to think at a higher-level of abstraction

- a *compiler* is a program that translates high-level code into machine code

  input: C++ language program
  output: machine language program

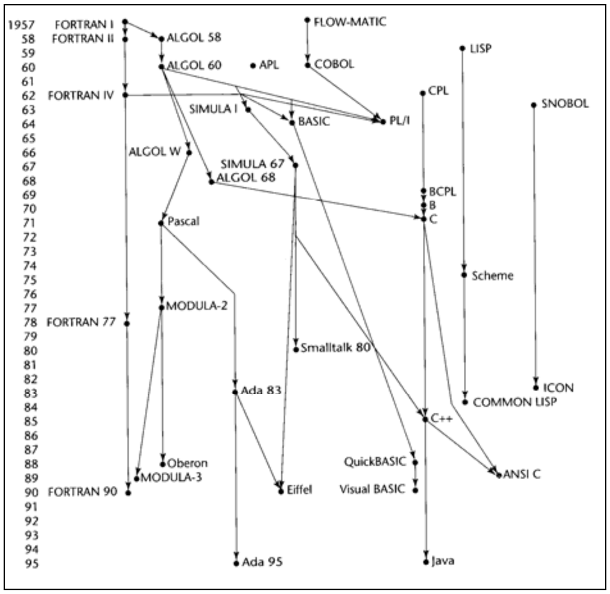
  similar to assembler, but more complex

```
/**
 * This class can print "Hello world!"
 *   @author Dave Reed
 *   @version 8/20/05
 **/

class Greeter
{
    public Greeter() { }

    public void SayHello() {
      System.out.println("Hello world!");
    }
}
```

- an interpreter is a program that reads and executes each language statement in sequence

  Java programs are first compiled into a virtual machine language (Java byte code) then the byte code is executed by an interpreter (Java Virtual Machine)

Why Java?



Java is a general-purpose, object-oriented language

- derived from C++, which was an object-oriented extension of C
- Java was designed to be a simpler, more robust language
- added features to make software engineering easier; removed features that led to confusion

Java and C++ are the dominant languages in industry

15

If you want to know more…

check out the following (purely optional) links

[Inventors: The History of Computers]

[Computer Museum History Center]

[Transistorized! from PBS.org]

[Apple Computer Reading List]

[The History of Microsoft]

[Internet Pioneers: Tim Berners-Lee]

[Internet Pioneers: Marc Andreessen]

[Wikipedia entry on Programming Languages]

[Webopedia entry on Programming Languages]

16

Exploring objects with Alice

Alice is a simple environment for creating and viewing 3-D animations

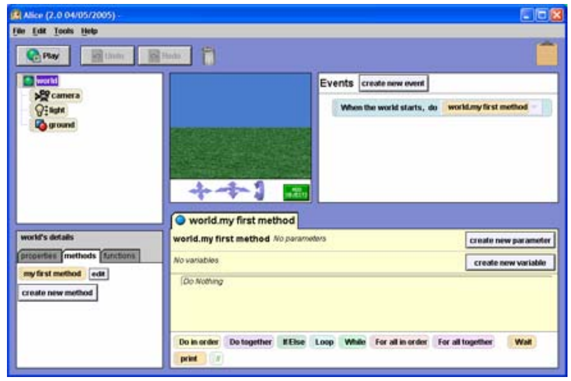
- developed at Carnegie Melon University for teaching introductory programming
- great for making object-oriented concepts concrete

  class: blueprint for a type of figure

  object: a particular figure in the scene (i.e., an instance of a class)

  fields: properties of an object

  methods: actions that the object can perform

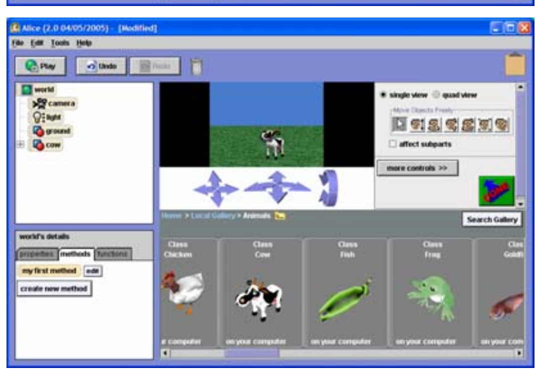


when you start up Alice, you get a blank scene (perhaps some default prompts the first time)

17

Alice

- click on the "Add Object" button to see a menu of figure types (i.e., classes) across the bottom
- you can click on any category, then select a figure type (i.e., class) and drag it onto the scene above
- you can resize/reorient this figure (i.e., object) using the buttons at the top-right
- you can add and position multiple figures (i.e.,objects) in the scene, even multiple instances of the same type (i.e., objects of the same class)
- click "Done" when the scene is set



Alice

- each figure (i.e., object) is composed of many smaller figures (i.e., objects)
- you can inspect the composite structure of a figure (i.e., object) in the upper-left pane

- each figure (i.e., object) has properties (i.e., fields) and predefined actions (i.e., methods)
- you can inspect these in the lower-left pane

- to perform an action (i.e., call a method), drag its box into the lower-right pane and select values (i.e., parameters) when prompted
- then, click on the "Play" button

Alice

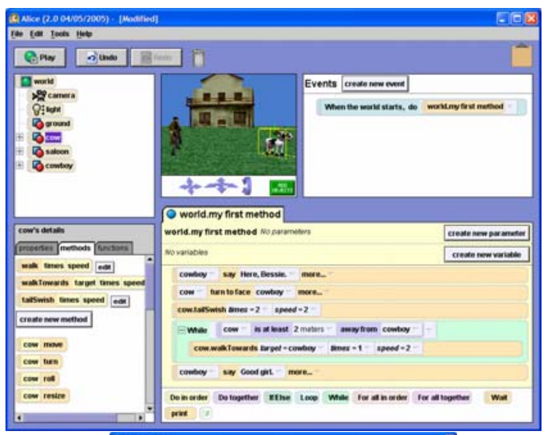
- you can drag a sequence of actions (i.e., method calls) into the lower-right pane to produce complex animations

- at the bottom of the pane, are drag-and-drop "control statements"

  Do-together: allows you to group actions (i.e., method calls) and perform them simultaneously

  Loop: allows you to perform an action (i.e., method call) a specified number of times

  If/Else and *While:* allow for conditional actions (i.e., method calls based on some condition) – must use functions from the lower-left pane for the condition



20