Abstraction

The Key to Managing Complex Processes

Notes for CSC 100 - The Beauty and Joy of Computing
The University of North Carolina at Greensboro

Reminders: What you should be doing!

Before Lab on Friday:

- Do Pre-Lab work for Lab 3

Reading:

- Make sure you participate in online discussion of *Blown to Bits* Chapter 1 (within the next week)

Class Exercise

In groups of 3-5 students:

Make a list of steps you take in the morning from waking up to being ready to go to school or work.

Obviously everyone might do things a little differently, but come up with a sequence of steps you can all agree on.

Forms of Abstraction

Descriptions and Example from Dan Garcia, UC Berkeley

- Detail removal

  "The act or process of leaving out of consideration one or more properties of a complex object so as to attend to others."

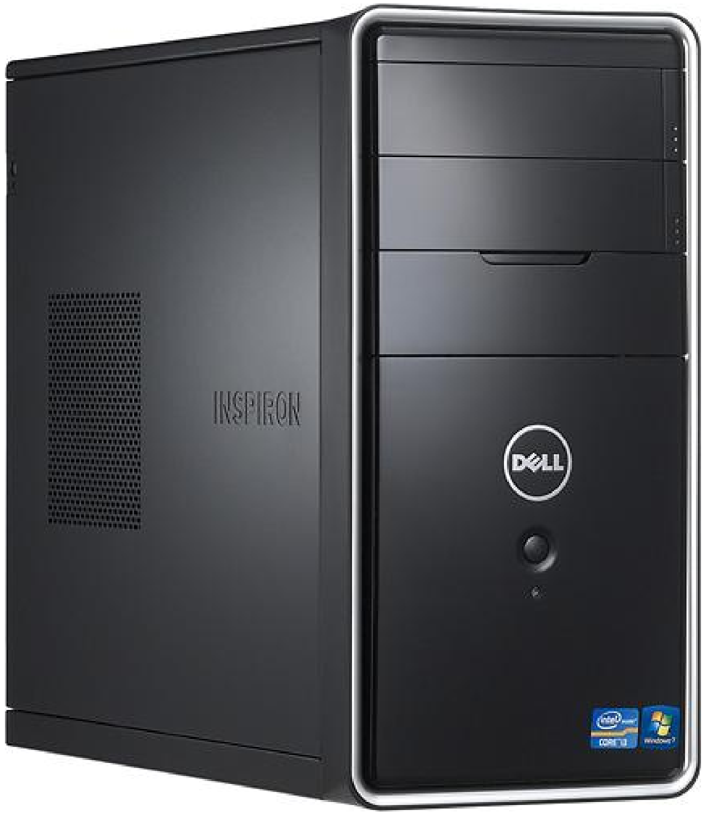
- Generalization

  "The process of formulating general concepts by abstracting common properties of instances."

Henri Matisse "Naked Blue IV"

From Dan Garcia, UC Berkeley

Question: What is this?



Detail Removal Example

Possible answers to previous question

A detailed answer:

A Dell Insprion Desktop, model I620-1996BK, with a 3.3 GHz Intel i3 processor, 4 GB or RAM, 500 GB 7200 rpm hard disk, Intel HD Graphics 2000, USB optimal mouse, and pre-installed with Windows 7 Home Premium (64 bit).

Just a few important technical details:

A Dell Inspiron Desktop with 4 GB of RAM and 500 GB hard disk.

The most basic description:

A computer.

Important point: Different levels of detail are suitable in different situations. An office designer doesn't need to think of this as anything other than "a computer" that needs to be placed in the room - details are superfluous and distract from what the designer is trying to do!

Detail Removal

A programming example

A program exists in many different levels of detail:

A high-level language (e.g., C++):

```
x = (y + 4) * z - 3;
```

Compiler produces...

Assembly language (readable but detailed):

```
movl  -4(%ebp), %eax
addl  $4, %eax
imull -8(%ebp), %eax
subl  $3, %eax
movl  %eax, -12(%ebp)
```

Aren't you glad you don't have to deal with this just to create a program?

Assembler produces...

Machine language (what is really executed):

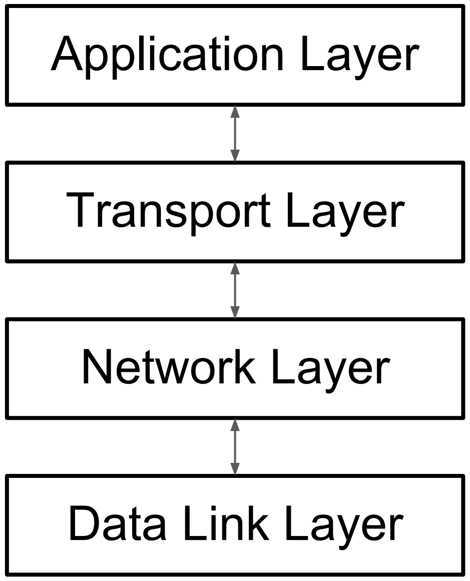
```
01000101 11111100 10000011 11000000 00000100
00001111 10101111 01000101 11111000 10000011
11101000 00000011 10001001 01000101 11110100
...
```

Question: If automated tools do this translation, why are multiple layers of abstraction useful?

Another layering example

Simplified network model (OSI model has 7 layers).

Each layer interacts with the one below it which has is less capable (less abstraction) than the one above.



"I want to retrieve the web page at http://www.google.com/"

"I want to connect to 74.125.136.105 and create a channel to send and receive bytes."

"I want to send this small packet of bytes to 74.125.136.105"

"I want to send this small packet of bytes to this other computer that I am directly connected to."

Some Quotes

Recall quote from Alan Perlis from last lecture:

> “A programming language is low level when its programs require attention to the irrelevant.”

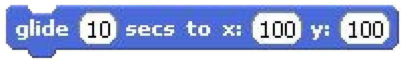
Another quote from Alfred North Whitehead (famous mathematician and philosopher from the early 1900’s):

> “Relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect increases the mental power of the race.”

Detail Removal

Snap! Example

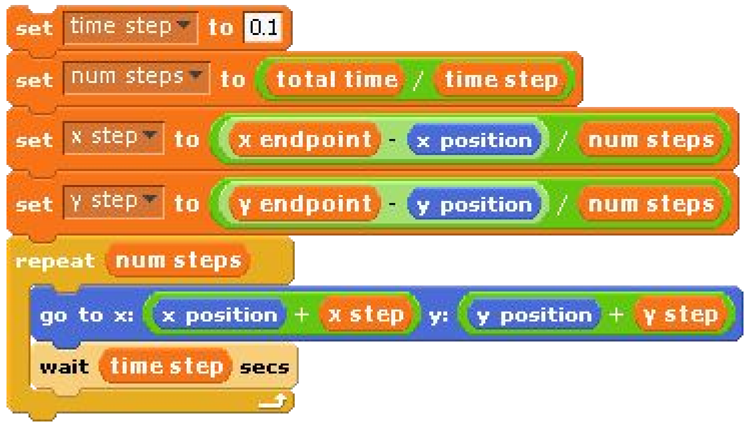
This:



```
glide 10 secs to x: 100 y: 100
```

Really does something like this:

(and even that is simplified...)



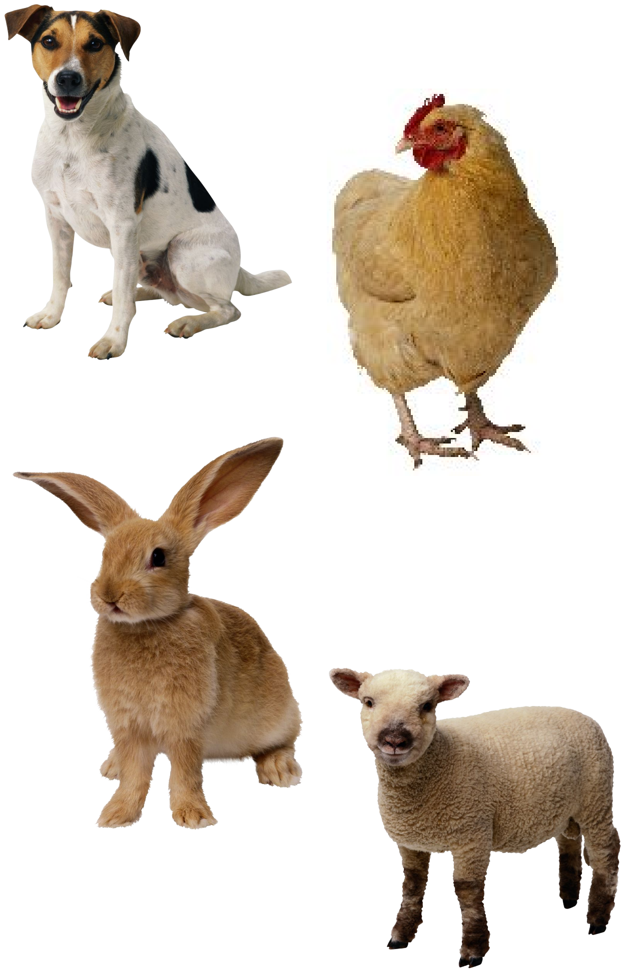
```
set time step to 0.1
set num steps to (total time / time step)
set x step to ((x endpoint - x position) / num steps)
set y step to ((y endpoint - y position) / num steps)
repeat num steps
    go to x: (x position + x step) y: (y position + y step)
    wait time step secs
```

Are the blocks provided by Snap! the only abstractions you will ever need?

NO! In this week's lab we'll see how to define our own blocks to make our own abstractions!

Generalization Example

- You have a farm with many kinds of animals
- Different food for each
- You have directions that say
  - To feed dog, put dog food in dog dish
  - To feed chicken, put chicken food in chicken dish
  - To feed rabbit, put rabbit food in rabbit dish
  - ...
- How could you do better?
  - To feed <animal>, put <animal> food in <animal> dish

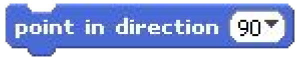


From Dan Garcia, UC Berkeley

Generalization in Programming

Snap! example

Think about this block:



Snap! could have provided a block that just pointed up...

- and one that just pointed down...
- and one that just pointed right...
- and one that just pointed left...

Instead have one generalized block, which is

- easy to think about and use,
- less worry after the initial development effort, and
- more powerful (can point at any angle).

Examples in Snap!

As time allows, we will spend the rest of the class doing examples in Snap!