COMP 1010- Summer 2015 (A01)

Jim (James) Young

young@cs.umanitoba.ca

jimyoung.ca

Website and labs

programmer-computer game

a) make pairs!!!

b) decide who is the programmer and who is the computer

task: write a set of instructions to reproduce a diagram:

no pictures! words only!

no metaphors or simile, no references, just basic objects and shapes!

programmer

computer

computers! put your heads down!!

computer

programmers, write your programs!

programmer

computers, execute the program!

computer

programming – not easy

and in this case, you had a smart HUMAN reading your program….

introduction to Processing (and Java)

(and some quick history)

computers work on just a bunch of switches!

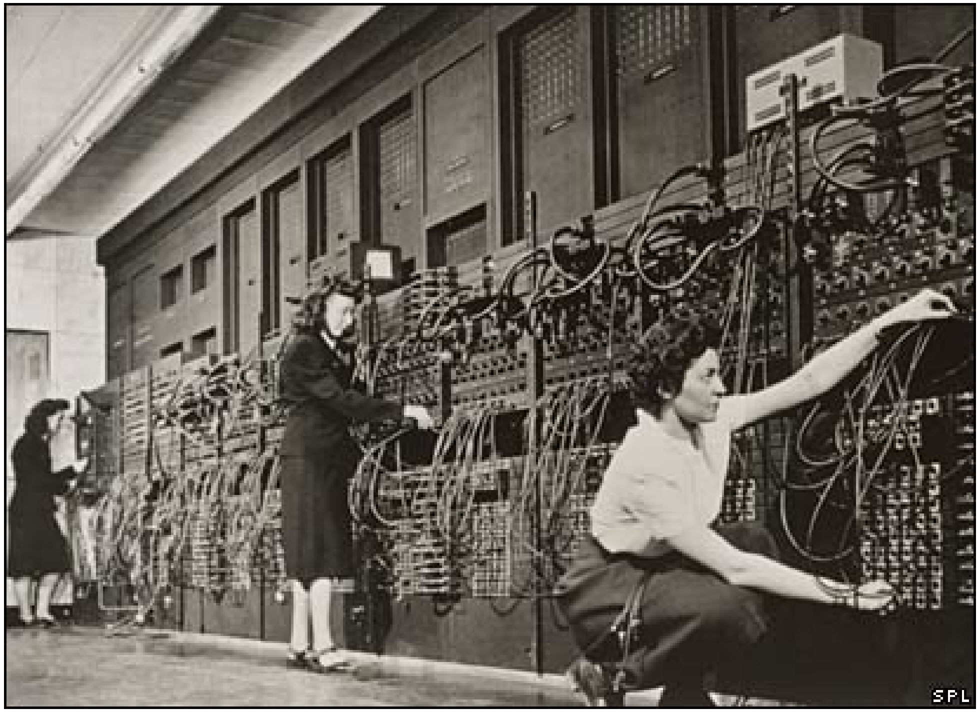
each switch can be either On or Off (no in the middle!)

shorthand, we say on is “1” and off is “0”

put 10 switches in a row: 0011011100

this language of 1s and 0s is called **binary**, and in the old days (<1950s), you would program a computer with a bunch of switches!

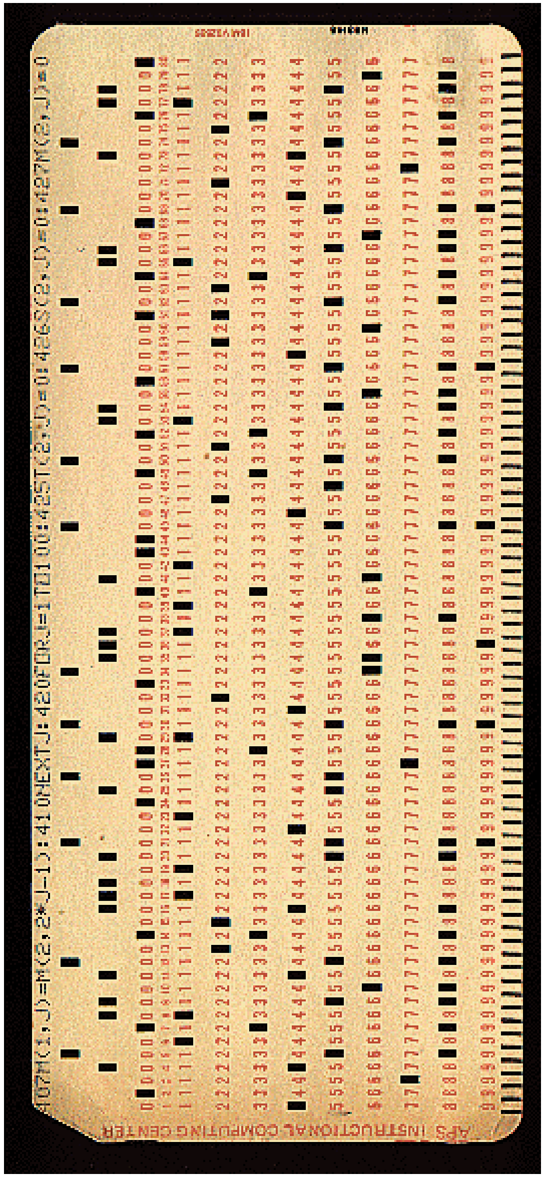
early programming..



eniac computer, 1946

5 million hand-soldered joints,
27tonnes, 150kWatts of power

5khz (5000 cycles / second),
385 multiplications / second



```
407M(1,J)=M(2,2*J-1):410NEXTJ:420FORJ=1TO100:425T(2,J)=0:426S(2,J)=0:427M(2,J)=0
```

punch cards: still binary
used even up to mid 70s

away from binary

assembly language: replace common binary sequences with commands for keyboard entry

e.g., (made up)

00111101 perhaps means to add the following numbers, so replace with the word “add”

this introduced a level of abstraction that made programming easier. This is “higher level” than pure binary programming

```
        ORG    ROM+$0000 BEGIN MONITOR
START   LDS    #STACK

***************************************
* FUNCTION: INITA - Initialize ACIA
* INPUT: none
* OUTPUT: none
* CALLS: none
* DESTROYS: acc A

RESETA  EQU    %00010011
CTLREG  EQU    %00010001

INITA   LDA A  #RESETA   RESET ACIA
        STA A  ACIA
        LDA A  #CTLREG   SET 8 BITS AND 2 STOP
        STA A  ACIA

        JMP    SIGNON    GO TO START OF MONITOR

***************************************
* FUNCTION: INCH - Input character
* INPUT: none
* OUTPUT: char in acc A
* DESTROYS: acc A
* CALLS: none
* DESCRIPTION: Gets 1 character from terminal

INCH    LDA A  ACIA      GET STATUS
        ASR A            SHIFT RDRF FLAG INTO CARRY
        BCC    INCH      RECIEVE NOT READY
        LDA A  ACIA+1    GET CHAR
        AND A  #$7F      MASK PARITY
        JMP    OUTCH     ECHO & RTS
```

High level languages

most programming languages aim to be **human-readable** (hah) so you don’t have to work in binary or assembly

java and processing are such “high-level” languages

introduction to Java

Java is a programming language
  a standard way to give
  instructions to a computer

Other languages:
  Perl
  Python
  C, C++
  C#

A basic Java program:

```
public class D1HelloWorld
{
        public static void main (String [] args)
        {
                System.out.println ("Hi out there!");
        }
}
```

Puts the text “Hi out there!” to the screen

A basic Processing program

```
line(0,0,10,10);
```

draws a line to the screen

introduction to processing

download it!

https://processing.org/download/

run it during class!!

You will download and use it in your first lab.

Processing Development Environment

- Project name
- Version
- JAVA processor
- Run and stop buttons. Shortcuts.
- Tabs (not used for us)
- Text editor.
- Message bar
- console

processing sketch

Programs are called Sketches!

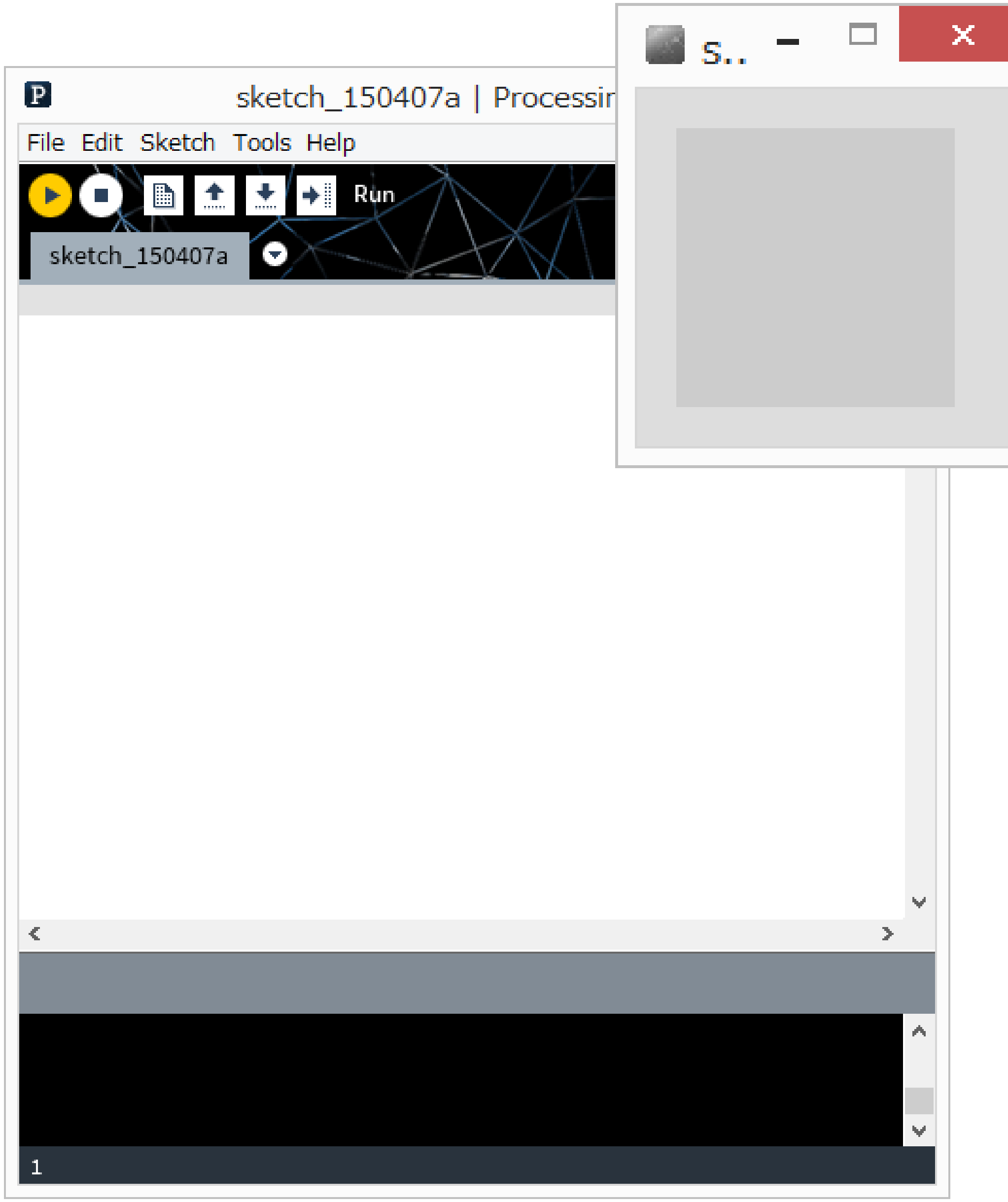
- Sketchbook
- Change sketchbook location
    - (BACKUP!!)

The canvas!!!

play button

stop button

grey inside and outside

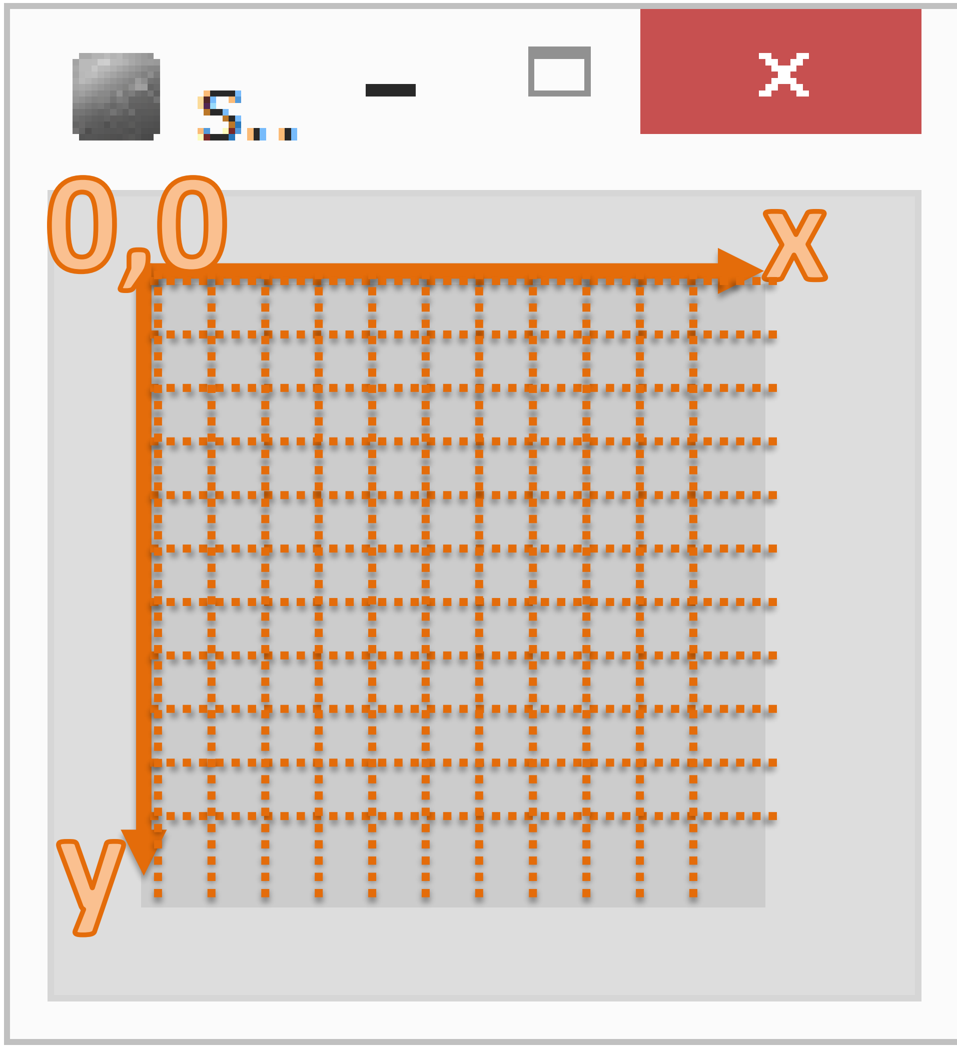


The Canvas co-ordinate system

Euclidean coordinates

0,0 at top left

Default window is 100 by 100 dots (pixels)



OK! Your first Processing Command draw a line!!!

Processing has some basic rules that we have to follow to give it commands. These rules are called **syntax**.

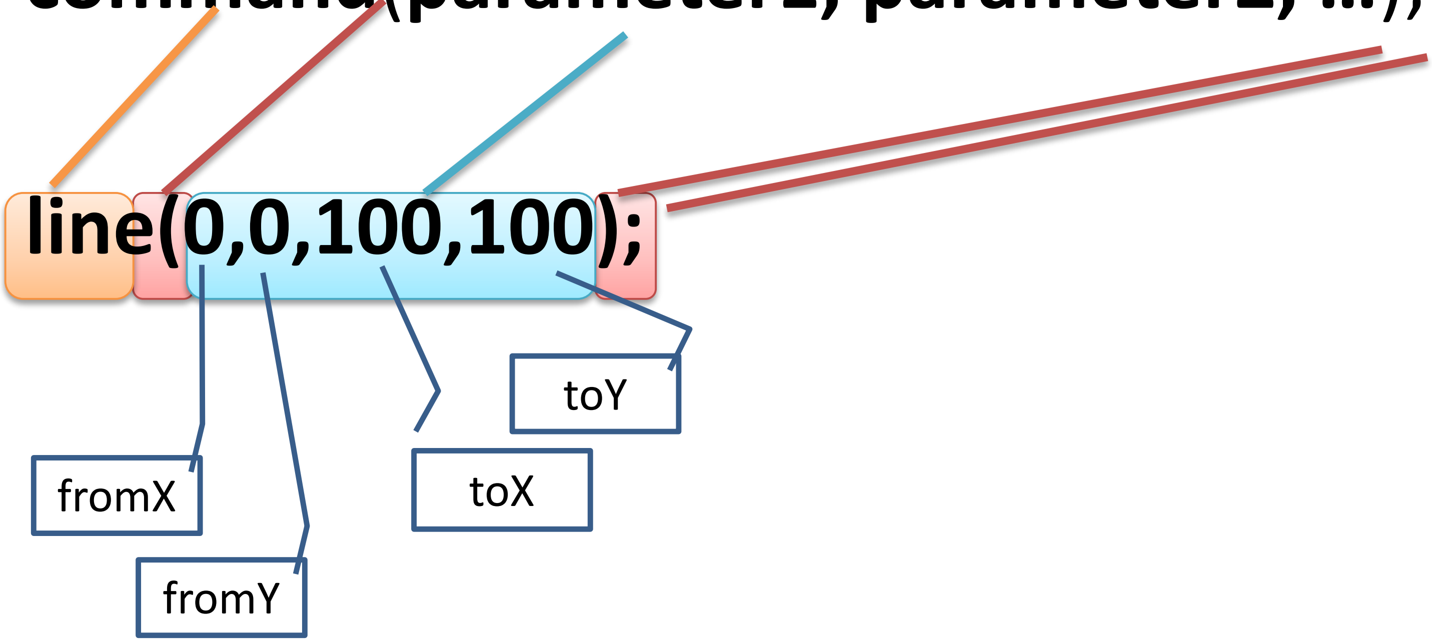
Remember how stupid computers are. We need to follow syntax or the computer won’t understand.

syntax of a command **statement**:

We need to specify the command (e.g, draw a line)

We need to pass information to the command (e.g., where to draw the line)

command(parameter1, parameter2, …);



line(0,0,100,100);

line(0,0,100,100);

How do you know what the required parameters are and what they mean???

Processing Help -> Reference

syntax issues…

command(**parameter**);

Everyone seems to remember the parenthesis ()

But **<u>don’t forget the semi colon!</u>**

The semi-colon means: end of command

Also – you can have spaces around the brackets
and commas – doesn’t matter.

What happens if you forget semi colon?

Processing is **case sensitive!**

anything you type in processing is **case sensitive**

this means that **upper case** is different than **lower case**

line ✓

Line ✗

LiNe ✗

(computers are stupid)

compile time errors

When Syntax Errors happen, Processing cannot convert your program into computer code. It happens when Processing tries to “compile” the program into binary. This means your program is never run.

some more processing commands

Check these out in the reference

size()

ellipse()

… at home

point()

rect()

triangle()