01 Introduction

Learning Processing Resources

Moodle website with lecture notes and lecture examples:
http://elearn.waikato.ac.nz/course/view.php?id=20984

Processing download: http://processing.org/download/

Processing Language Reference: http://processing.org/reference/

Online Tutorials: http://processing.org/tutorials/

Online Examples: http://processing.org/examples/

Processing for Javascript: http://processingjs.org/

Setting up Android Mode: http://processing.org/tutorials/android/

Community site: http://openprocessing.org/

Basic Processing Concepts

Sketch – a window or screen where your program displays graphics and images with its drawing commands.

Frame – an update to the sketch, usually occurs every $1/60^{th}$ or $1/30^{th}$ of a second.

Setup() – a special function called once at the start of the program to set everything up.

Draw() – another special function called once per frame, used to update the screen.

Coordinate System – (0,0) is defined at the top left of the sketch, *x* increases as you move right, *y* increases as you move down.

Built-In Variables – Processing defines several very useful built-in variables, e.g. `width`, `height`, `frameRate`, `mouseX`, `mouseY` etc.

Basic Programming Concepts

Instruction Sequence – Processing is a Java-dialect, very similar to C#; semi-colons separate statements and curly braces define code blocks.

Primitive Data Types – `int`, `float`, `char`, `String`,`boolean`, `color`.

```
int length=43;
float x=23.3, y;          // y is created but uninitialised
char firstLetter='M';
String name="John";
int m = (int) x;          // what value does m have?
boolean result1=true,
boolean result2=(length<=40); // what is result2?
```

Color Data Type – specifies an RGB color, created using either the color() function, e.g. `color(231, 67, 88);` or hexadecimal notation, e.g. `#88A2FA.` The range of colour values is usually 0 (darkest) to 255 (brightest).

```
color appleGreen = color(17,191,18);
color white = color(255);
color purple = #BF11B6;
```

Arrays – must be created with the `new` keyword and use square bracket notation for assignment, e.g.

```
int[] sizes=new int[10];
sizes[3]=45;
```

Scope – variables declared inside functions are local, variables outside all functions are global.

```
int x=10,y=11;
void setup() {
   int x=12;
   println(x);
   println(y);
}
```

Conditional Statements – if/then/else, switch.

```
if (x>width) {
   println("x is too large");
   x=0;
} else
   x++;

int value=6;
switch(value){
   case 1: println("1"); break;
   case 2: println("2"); break;
   case 6: println("6"); break;
   default: println("unknown value");
}
```

Iteration – for loops, while/do loops, do/while loops.

```
for (int index=0; index<size; index++) {
   println(index);
}

int index=0;
while(index<10) {
     println(index);
     index++;
}
```

Iteration over arrays – special form of a for loop exists

```
int[] ages = {34,56,12,88};
void setup() {
  for (int age: ages) {
    println(age);
  }
}
```

Functions – Processing supports functions that either don't return a value (void functions) or do return a value.

```
String concat(String first, String last){
  String result=first;
  result+=" ";
  result+=last;
  return result;
}
void debug() {
  println( concat( "John", "Smith" ) );
  println( concat( "Amy", "Pond" ) );
}
void setup(){
  debug();
}
```

Constants – use the final keyword to indicate that a value is a constant

```
final int NUM_LEVELS = 8;
```

Principles of Good Programming

Do not duplicate code – use arrays, loops and functions to eliminate unnecessary duplication.

Example beginner code:

```
int x1 = 10;
int x2 = 20;
int x3 = 30;
int x4 = 40;
…
int x30 = 300;
```

Professional code:

```
final int NUM_ITEMS = 30;
int[] xPositions = new int[ NUM_ITEMS ];
for (int index=0; index<xPositions.length; index++)
     xPositions[ index ] = (index+1) * NUM_ITEMS;
```

Use proper naming convention – `lowerCamelCase` for variable and function names; `UpperCamelCase` for class names; `FULL_UNDERSCORED_CAPS` for constants. Use lengthy, descriptive names wherever possible except where the meaning is obvious.

Comment your code – Processing supports `//` for single line comments and `/*...*/` for multi-line comments. Write descriptive comments so that your peers can understand your program.

```
// Calculate the average age of participants
float averageAge=0;
for (int age: ages)      // iterate over ages array
     averageAge+=age;
averageAge /= ages.length;
```

```
/*
 * Survey
 * Author: J. Smith.
 *
 * A program to prompt users for demographics details
 * (age, gender etc) and then ask them a series of
 * survey questions.
 */
```