

LCC 6310 The Computer as an Expressive Medium

Lecture 2

Administrivia

Book(s)?

Processing installed?

Lab times

Friday 9:05 - 10:55, IDT Lab

TA: Micah Horvat, mhorvat3@gatech.edu

Overview

Discuss variables & if-then

Discuss readings

Look at assignment 1

Readings for next week

Drawing primitives revisited

What are the different parts of a drawing primitive?

The diagram shows the code snippet `line(0, 0, 50, 50);` with several annotations. A green arrow points from the text "method name" to the word "line". A blue arrow points from the text "arguments" to the four numbers "0, 0, 50, 50". A pink arrow points from the text "parentheses contain arguments" to the opening and closing parentheses of the code.

```
line(0, 0, 50, 50);
```

method name

arguments

parentheses contain arguments

The drawing commands are *methods*

Methods are reusable commands

Like a little machine that does work for you

Let's you reuse code without typing it over and over

The arguments tell the method precisely what to do

We'll see later that you can define your own methods!

Introduction to variables

A variable is a named box for storing values

You can put values in a variable by using the assignment operator (in Java, the assignment operator is "=")

e.g. `x = 1;`

To use the value stored in a variable, just use the variable's name

e.g. `line(x, 0, 50, 50);`

But how does Java know what kind of value can go in a variable?

Variables have a *type*

You must tell Processing (i.e. Java) what kinds of values can go in the box

You do this by giving a variable a *type*

```
int x; // variable x can hold integers (int)
int y; // variable y can hold integers (int)

x = 20; // store 20 in x
y = 30; // store 30 in y
point(x, y); // use the values of x and y to draw a point
```

Let's try it in [Processing](#)

Effect of creating an int variable

Code

```
// Single int
int anInt;
```

```
// Put a value in the int
anInt = 3;
```

```
// Type error!
anInt = "hello";
```

Effect

Name: `anInt`, Type: `int`



Name: `anInt`, Type: `int`



Name: `anInt`, Type: `int`



Can't shove "hello" into an int

Assigned values must match the type

```
int x; // variable x can hold integers (int)
int y; // variable y can hold integers (int)

x = 1.5; // store 1.5 in x causes an error!!!
y = 30; // store 30 in y
point(x, y); // use the values of x and y to draw a point
```

Let's see this in [Processing](#)

Can combine declaring and assigning

Declaring a variable means telling Processing it's type

```
int x; // variable called 'x' is an integer
```

Assigning a value to a variable means putting a value in the named box

```
x = 1; // variable called 'x' stores value 1
```

You can declare and assign at the same time

```
int x = 1;
```

But only declare a variable once, otherwise you get an error!

The "primitive" types

int – integers between -2,147,483,648 and 2,147,483,647

That's pretty big! (32 bits or 4 bytes long to be precise)

float – floating point numbers (e.g. 3.1415927, -2.14)

These are 32 bits long as well

char – a single character (e.g. 'c')

byte – integers between -128 and 127 (i.e. 8 bits)

boolean – holds the values 'true' or 'false'

color – holds a color (red, green, blue, alpha)

```
color c1 = color(255,0,0, 0); // pure red color, transparent
```

```
color c2 = color(255,0,0,255); // pure red color, opaque
```

Let's try these in [Processing](#) and find them in the [reference](#).

Print and println

When working with variables, it is often useful to look at their values

print() and println() print to the bottom processing pane

They do the same thing, except println starts a new line after printing

Let's try this out in [Processing](#)...

Control flow

By default Java executes the lines of a method one after the other

Sequential control flow

Unconditional – doesn't matter what happens in the world

Often we want to control which steps are executed depending on what else has happened

That is, we want *conditional* control flow

This is necessary in order to make anything that is *interactive*

if

`if` statements introduce conditional branches

```
if (<boolean expression>){  
    // do this code  
}
```

Boolean expressions have one of two values: `true` or `false`

if-else variation

```
if (<boolean expression>){  
    // do this code  
} else {  
    // do this other code  
}
```

if-elseif variation

```
if (<boolean expression>){  
    // do this code  
} else if (<another boolean expression>) {  
    // do this other code  
}
```

Some boolean expressions

`anInt == 1` true if the value of variable `anInt` is equal to 1

`x > 20` true if the value of variable `x` is greater than 20

`1 == 2` true if 1 is equal to 2 (it's not so this is false)

`!` is the not operator – reverses true and false so,

`!(1 == 2)` is true!

This is not a boolean expression

```
int anInteger = 1;
```

Remember, '=' is the assignment operator, while '==' is used to check if the two sides of the expression are equal to one another

Example

```
strokeWeight(2); // draw with heavier line
stroke(200, 0, 0); // draw with redish line
```

```
boolean drawRect = true;
boolean drawAnX = true;
```

```
if (drawRect) {
    fill(0, 200, 0); // fill with green
    rect(30, 30, 40, 40);
}
if (drawAnX) {
    line(0, 0, 100, 100);
    line(100, 0, 0, 100);
}
```

Try changing the values of `drawRect` and `drawAnX`

Readings

Summary presentations & questions for discussion

From Software: Exhibition at the Jewish Museum (NMR p.247)

Four Selections by Experiments in Art and Technology (NMR p.210)

Concepts, Notations, Software, Art (Linked from course website)

What do you think of this?

"Which means that those computer scientists who invented these technologies ... are the important artists of our time, maybe the only artists who are truly important and who will be remembered from this historical period."

Lev Manovich

Why program?

Relationship between art and CS

Should new media artists program? Why?

What about collaboration?

Computation as a medium

Assignment 1

Posted [online](#), due **Friday August 31st**

- A1-01: Draw three lines.
- A1-02: Draw five lines.
- A1-03: Draw three ellipses.
- A1-04: Control the position of two lines with one variable.
- A1-05: Control the position and size of two lines with two variables.
- A1-06: Control the properties of two shapes with two variables.
- A1-07: Create a simple, regular pattern with six lines.
- A1-08: Program your pattern from Assignment 1-07 using while().
- A1-09: Draw a layered form with two new loops.
- A1-10: Redo Assignment 1-05 using mouseX and mouseY as the variables.
- A1-11: Draw two visual elements that each move in relation to the mouse in a different way.
- A1-12: Draw three visual elements that each move in relation to the mouse in a different way.
- A1-13: Move a visual element across the screen. When it disappears off the edge, move it back into the frame.
- A1-14: Draw a visual element that moves in relation to the mouse, but with a different relation when the mouse is pressed.
- A1-15: Using if and else, make the mouse perform different actions when in different parts of the window.
- A1-16: Develop a kinetic image which responds to the mouse.

Readings for next week

For **Tuesday** next week: Java Readings

Concepts: variables & arrays

Processing, pp. 83-85

For **Thursday** next week: Theory Readings

Two students: present one reading each

Everyone else: prepare one discussion question for each reading

Man-Computer Symbiosis - J.C.R. Licklider (NMR p.73)

Personal Dynamic Media - Alan Kay & Adele Goldberg (NMR p.391)