Diploma in Computer Studies
Sep 2021

Welcome to Creative Computing

DCR2284

Learning Objectives

- At the end of the course, students will be able to:
  - CO1: Describe the creative concepts in mathematics and computing.
  - CO2: Explain the importance origins of geometry to develop motion, images and sound.
  - CO3: Build the Processing application to construct shapes and objects.
  - CO4: Write the coordinate transformations for motions using Processing..

Revisit First Week!

- Introducing Creative Computation
- Introducing ourselves
- Introducing Processing
- Drawing
- Living code

Overview

- Introducing Creative Computation
- Introducing ourselves
- Introducing Processing
- Drawing
- Living code

Variables (2-a) and conditionals (2-b)

Variables
or
Numbers

Enter the variable

Numbers are too permanent

```
size(500,500);
rect(200,200,100,200);
rect(225,150,50,50);
rect(200,400,25,100);
rect(275,400,25,100);
```

- Which of these rects are the legs?
- How do I change the lengths of the legs?
- How do I put my avatar somewhere else?

Numbers are too permanent

```
size(500,500);
rect(200,200,100,200);
rect(225,150,50,50);
rect(200,400,25,100);
rect(275,400,25,100);
```

- Which of these rects are the legs?
- How do I change the lengths of the legs?
- How do I put my avatar somewhere else?

Hardcoded numbers are a huge pain to edit and are mostly meaningless to look at in code

Numbers can't change

```
size(500,500);
rect(200,200,100,200);
rect(225,150,50,50);
rect(200,400,25,100);
rect(275,400,25,100);
```

- How would I make my avatar move around on the screen?

Numbers can't change

```
size(500,500);
rect(200,200,100,200);
rect(225,150,50,50);
rect(200,400,25,100);
rect(275,400,25,100);
```

- How would I make my avatar move around on the screen?
- Yeah, I'd need to change the *numbers* representing its location, but I *can't*, because...

Numbers can't change

```
size(500,500);
rect(200,200,100,200);
rect(225,150,50,50);
rect(200,400,25,100);
rect(275,400,25,100);
```

- How would I make my avatar move around on the screen?
- Yeah, I'd need to change the *numbers* representing its location, but I *can't*, because...

Once the program is running, hardcoded numbers cannot change

Enter the variable

A variable is like an... x

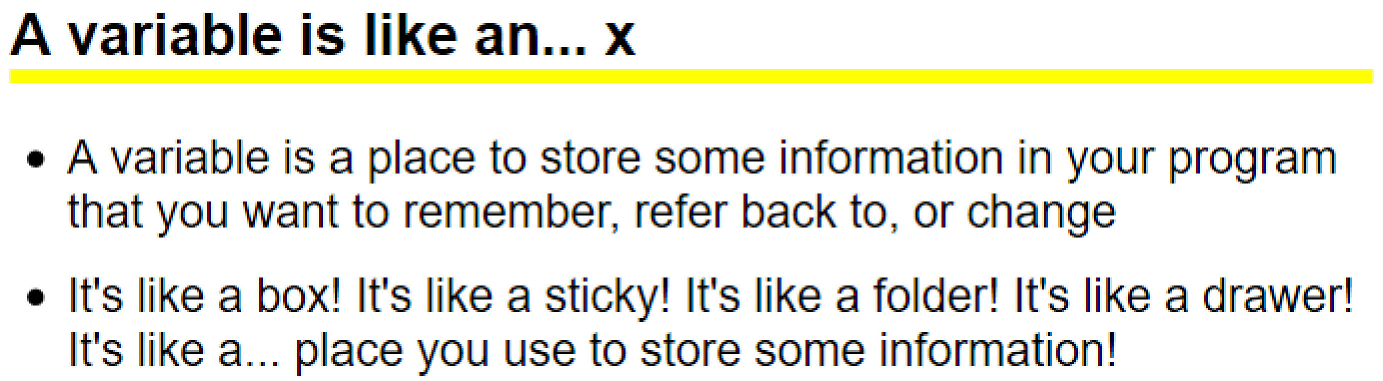
- A variable is a place to store some information in your program that you want to remember, refer back to, or change
- It's like a box!

A variable is like an... x

- A variable is a place to store some information in your program that you want to remember, refer back to, or change
- It's like a box! It's like a sticky!

A variable is like an... x

- A variable is a place to store some information in your program that you want to remember, refer back to, or change
- It's like a box! It's like a sticky! It's like a folder!



A variable is like an... x

- A variable is a place to store some information in your program that you want to remember, refer back to, or change
- It's like a box! It's like a sticky! It's like a folder! It's like a drawer! It's like a... place you use to store some information!

A variable is like an... x

- A variable is a place to store some information in your program that you want to remember, refer back to, or change
- It's like a box! It's like a sticky! It's like a folder! It's like a drawer! It's like a... place you use to store some information!
- It has three key qualities:

A variable is like an... x

- A variable is a place to store some information in your program that you want to remember, refer back to, or change
- It's like a box! It's like a sticky! It's like a folder! It's like a drawer! It's like a... place you use to store some information!
- It has three key qualities: a *name*, a *type*, and a *value*

A name

- Variables have names - otherwise what would you call them?
- Names have *meanings*. "Thomas" means "twin" (via Aramaic), "Jeanne" means "Yahweh is merciful" (via Hebrew)

A name

- Variables have names - otherwise what would you call them?
- Names have *meanings*. "Thomas" means "twin" (via Aramaic), "Jeanne" means "Yahweh is merciful" (via Hebrew)
- `mouseX` means "the pixel on the x-axis that the mouse is over right now"

A name

- Variables have names - otherwise what would you call them?
- Names have *meanings*. "Thomas" means "twin" (via Aramaic), "Jeanne" means "Yahweh is merciful" (via Hebrew)
- `mouseX` means "the pixel on the x-axis that the mouse is over right now"
- `avatarY` probably means "the location of the avatar on the y-axis"

A type

- In Processing (Java) you need to say what kind or *type* of value a variable has in it
- Like a number or a colour or a string of letters for example
- This is so the language knows what kinds of things you can do with it
- You don't multiply a letter by a number, for instance

A value

- A variable has a value in it, the thing that variable is storing
- The variable will keep it safe until we need to remember it, use it, or change it
- And a change is as good as a holiday

`int meaningOfLife = 42;`

- This is a *variable declaration* in Processing
- This line is us telling Processing "I want a *variable* to store an *integer* called *meaningOfLife*, and put the integer `42` in it to start with, thanks"
- Let's go through the pieces of this

```
int meaningOfLife = 42;
```

- First we write the *type* of the variable
- In this case we want an *integer*, which is abbreviated to `int` in programming
- An integer is a whole number that can be positive, negative, or zero

```
int meaningOfLife = 42;
```

- Next we have the *name* of the variable
- Here we have called it `meaningOfLife`
- Just like with functions, the name should *explain* what the variable is for, what it *means*

- Notice the way the variable name is written
  - All one word - no spaces
  - If there are multiple words make the first lowercase and each next word starts with a capital
  - You can have numbers, but not at the start of the name

```
int meaningOfLife = 42;
```

- Next we have the *assignment operator*
- This is because we are giving our variable a value right away
- It means "I am about to tell you what to put inside this variable"
- Be ready for this use of an equals sign to be confusing when we need to actually check whether two things are equal!

`int meaningOfLife = 42;`

- Next we have the *value* of the variable, `42`
- Importantly, `42` is an `int` and it is also the `meaningOfLife`
- Note that because we said this variable is an `int` we're *not allowed* to put any other kind of value here
  - Can't have a number like `1.2345`
  - Can't have a string of characters like `"Hello, world!"`

```
int meaningOfLife = 42;
```

- Just like any line of code that is an *instruction*, we end with a semicolon to say we're done
- If you're a polite kind of person, you could think of it as saying "thanks" perhaps...

Other types

- Processing doesn't just have integer variables using `int`, there are other types too, like:

```
float piToThreeDecimalPlaces = 3.14;
char theCharacterA = 'a';
String helloWorld = "Hello, World!";
color red = color(255,0,0);
```

- Notice how the different values are written in different ways - a `float` can have a decimal point, a `char` is inside single quotes, a `String` is inside double quotes, a `color` requires that you use the *function* `color(r,g,b)` to create the value

Declaring a variable without a value

- We can also declare a variable we want to use in our program *without* giving it a value right away

```
int meaningOfLife;
```

- Later on, when we work out the meaning of life, we can use the *assignment operator* in the same way to put the value in

```
meaningOfLife = 42;
```

- Notice that when we put the value in we don't need `int` anymore, because Processing already *knows* it's an `int`

`println()` is pretty helpful

- Last week we used `println()` to print out "Hello, World!" on the console
- But we can also use it to print out the values of variables, which can be very helpful

```
int meaningOfLife = 42;
println(meaningOfLife);
```

`println()` is pretty helpful

- Last week we used `println()` to print out "Hello, World!" on the console
- But we can also use it to print out the values of variables, which can be very helpful

```
int meaningOfLife = 42;
println(meaningOfLife);
```

- Pop quiz: what would happen if we *didn't* give a value to `meaningOfLife` and then tried to `println()` it?

Using variables...

- You can use variables **as if they are the value inside them**
- So you can use an `int` variable anywhere you might use a hardcoded integer, a `String` variable anywhere you would have put a string, and so on!

```
int meaningOfLife = 42;
rect(meaningOfLife,meaningOfLife,50,50);

String helloWorld = "Hello, World!";
println(helloWorld);
```

Arithmetic!

- You can do arithmetic on numbers in Processing, and also on variables with numbers in them
- It uses symbols you probably already know from calculators and so on

```
int meaningOfLife = 21 + 21; // addition
println(meaningOfLife - 2); // subtraction
fill(meaningOfLife * 5,0,0); // multiplication
rect(meaningOfLife/2,0,50,50); // division
```

- There are other operators too, which you can look up in the reference
- What will the above code actually do?

More arithmetic!

- You can use parentheses to prioritise parts of your arithmetic, just like in math class...

```
int meaningOfLife = (42 + 42) / 2;
```

is not the same as

```
int meaningOfLife = 42 + 42 / 2;
```

Space, man

- Pay attention to spaces. A lot of the time they're not strictly necessary, but they make things a lot easier to read.
- These two are equivalent:

```
int meaningOfLife=(42+42)/2;
```

```
int meaningOfLife = (42 + 42) / 2;
```

- But the second one is easier to read, right?

Variable names, again

- Remember that there are *rules* for naming variables
- They *must* start with a letter and continue only with letters, numbers, or the underscore character _
- They *must* be unique and should not already be in use by Processing (e.g. not `mouseX`)
- They *must* be meaningful
- They *should* use "camel case" where you start with a lowercase letter and then use capital letters to indicate word breaks

Example variable names

YES:

```
int age = 30;
String dayOfTheWeek = "Friday";
float pi = 3.14159;
char theLetterE = 'E';
int theNumber1 = 1;
```

NO:

```
int foo = 30;
String WhatDayIsIt? = "Friday";
float 314159 = 3.14159;
int int = 1;
char the_letter_a = 'A';
```

Built-in variables

- We met `mouseX` and `mouseY` last week - they are *built-in variables* that store the current coordinates of the mouse

- There are other helpful variables like this, including:
    - `width` and `height`: the width and height of the window
    - `frameCount`: the number of frames the code has run for
    - `frameRate`: the frame-rate of the code
    - `mousePressed`: true if the mouse button is currently pressed down, false otherwise
    - `key` and `keyCode`: the most recently pressed key

Variables give us a lot of power in programming.

Memory. Now we can *remember* values over time.

Sense. Now we can *label* values with their meaning instead of hard-coding them.

Change. Now we can *change* the values in variables to make things happen while the program is running.

Variables save the day

Remember this guy?

```
size(500,500);
rect(200,200,100,200);
rect(225,150,50,50);
rect(200,400,25,100);
rect(275,400,25,100);
```

Now we can rewrite him with variables to get our benefits of memory, sense, and change...

```
int avatarX = 50;
int avatarY = 50;
int avatarHeadSize = 50;
int avatarBodyWidth = 100;
int avatarBodyHeight = 200;
int avatarLegWidth = 25;
int avatarLegHeight = 150;

void setup() {
  size(500,500);
}

void draw() {
  rect(avatarX, avatarY, avatarBodyWidth, avatarBodyHeight);
  rect(avatarX + (avatarBodyWidth - avatarHeadSize)/2, avatarY - avatarHeadSize,
       avatarHeadSize, avatarHeadSize);
  rect(avatarX, avatarY + avatarBodyHeight, avatarLegWidth, avatarLegHeight);
  rect(avatarX + avatarBodyWidth - avatarLegWidth, avatarY + avatarBodyHeight,
       avatarLegWidth, avatarLegHeight);
}
```

Let's get `random()`

- Let's talk about my favourite function in all of programming: `random()`
- Most programming languages have a version of this and it does what you might expect... yeah, it gives you a random number.

Let's get `random()`

- Let's talk about my favourite function in all of programming: `random()`
- Most programming languages have a version of this and it does what you might expect... yeah, it gives you a random number.
- In Processing it works like this:

```
float randomNumber = random(n);
```

- This will put a random `float` between 0 and `n` (not including `n`) into our `randomNumber` variable
- `random(10)` gives us a random floating point number between

Let's get more `random()`

- You can also specify a *range* for your random number like this

```
float red = random(200,255);
float green = random(0,100);
float blue = random(0,100);
background(red,green,blue);
```

- Which will do what?

Ah, `random()`!

- Random numbers are a source of endless joy.
- What would this do in the `draw()` loop of our avatar code?

```
avatarX = floor(random(0,width));
avatarY = floor(random(0,height));
```

- What is `floor()`, you ask? It's a math function that turns a `float` into an `int` by removing everything after the floating point
- So `4.5489549` becomes `4` and `145.1` becomes `145` etc.

One last amazing type!

- Before we move on I want to very briefly introduce another type
- It's called `PImage` and it is a special Processing type that can contain... an *image*!
- I want us to see it now because drawing things out of shapes can be amazing...
- ... but it can also be nice to use images as well (like people do in the real world)

`PImage`

To use a `PImage` we need to

1. Declare a `PImage` variable
2. Load the image file into the variable
3. Display the image like any old shape

Generally we do this the same way each time, so here's a template of it...

```
PImage myImage;

void setup() {
  size(500,500);
  myImage = loadImage("dog.png");
}

void draw() {
  image(myImage,0,0);
}
```

Note! You have to use `loadImage()` inside `setup()` or it won't work!

Note! Your image file (`dog.png` here) must be inside a folder called `data` inside your sketch folder!

Note! You can learn plenty more about `PImage` in... the reference under *Image*!

We will continue on Week 2 (b) on Conditionals (IF Statement)

Stretch Break!

Thank you

Workshop Lab

Introduction to Processing