From Maths To Code

CBP 15-03-20

How to transform maths expressions into code? Here are some examples.

1.

$$\tau_m = A_d \cos\left(\frac{2\pi t}{T_d}\right)$$

torqueM = driveAmp*cos(2*pi*time/drivePeriod);

so you see when a maths expression has two symbols next to each other, then you multiply them in code. Here's another example

2.

$$\tau = -mgLsin(\theta) - b_a\omega + \tau_m$$

torque = -mmass*gravity*armLength*sin(theta) – armDamp*omega + torque;

3. And finally we have the "dynamics":

$$\Delta\omega = \alpha\Delta t$$
$$\omega = \omega + \Delta\omega$$

These two lines of maths are coded as

```
deltaOmega = alpha*dT;
omega = omega + deltaOmega;
```

or as

```
omega += alpha*dT;
```

Coding Unreal Script for the MAS14 Engine

1. Using Variables

You should use global variables if you want so send your variable to the HUD or to log it for data analysis.
How to declare a global variable? These are declared at the top of the source using the syntax, e.g
var float name;

You should code *parameters* like this so you can change them in the editor
var(MAS14) float name;

Local variables are used within functions, and their values are lost when the function returns. They are used to hold intermediate results and are declared within a function like this
local float name;

2. Sending Data to the HUD

Add the following to the function **SendValuesToHUDX(…)** to put **myVar** on the HUD

HUDLine[N] = "myVar"@myVar;

where N is the next line number. You can specify a color by writing

LineCol[N} = MakeColor(R,G,B); where R,G,B are in the range 0-255

3. Sending Data to the log file

First add the following line of code to the function **setLogFileColumnLabels(…)** to set the label for your data column

columnLabels[N] = "myVar";

Then in the function **logDataRecord(…)** write the following line

dataArray[N] = myVar;

4. Assigning parameters to "setParam"
This happens in the **function setParam**(…). If you have a parameter **myParam** then this is how you assign it**.**

```
if(paramName == Caps("myParam")) {
  myParam = paramValue;
}
```

Repeat this as many times as you like.

5. Assigning function keys

This happens in the **function ProcessKey(…)** If you want to use a particular key to provide user input (to set the value of a variable, or to call a function) then do this

```
if(Key == 'a') {
  myVariable = myValue;
}
if(Key == 'b') {
  myFunctionCall();
}
```

6. Assigning values to your parameters and variables

This depends on whether you want to change values during the investigations, either using **setParam** during a run or else in the Editor before a run. If you wish to change variables then

(i) Declare them like this **var(MAS14) float myParam;**
(ii) Give your variable a default value like this **myVar=12.3** in the **defaultProperties** code block at the end of the script.

Some variables (especially those which change during computation) need initial values setting;

(i) Declare it like this **var float myVar;**
(ii) Initialise it **myVar = 10;** in the **initialiseVariables()** function.

7. Using Arrays

You can make arrays of most things; both primitives (int, float, bool, etc) or Actors (think "object"). You declare an array like this (for the case of a global array)

var array<float> myArray;

This array is *dynamic*, its length can change during execution.

The length of the array at any time is **myArray.length**

To add an element to your array on the fly, do this

myArray[myArray.length] = something;

This will increase the length of the array by 1.

To use the array in a loop then here's what to do.

```
local int i;
i=0;
while(i<myArray.length) {
  val = myArray[i];
  i++;
}
```