Loops

Technical Data

Introduction to Loops

Loops, how can I explain loops.... they have absolutely nothing to do with shoelaces or acrobatic aircraft. A loop, in scripting, is a piece of script that will run repeatedly until it is told to stop. Sort of like the enterframe event but with brakes.

Lets say I want to duplicate a movie clip 1,152 times, the script used to do this is reasonably easy we just need to write the duplicate script 1,152 times. Well easy but amazingly boring and time consuming. Although this is exactly the situation a loop loves... a lot. We will learn how to do this in the second part of this workshop. But here is an example of a loop that will run 1,152 times:

```
for(i = 1; i <= 1152; i++){

    // DO SOME STUFF

}
```

The example used above is a for loop. Inside the brackets of a for loop are three "arguments" and each of those arguments is seperated by a semi-colon ; . Below is explanation of what each argument does:

i=1 -- This is the first part of the loop, it sets the value of a variable (i) when the loop first starts.

In this case the variable is called i. The letter i is recognised as a variable that is used to hold numbers that are counting either up or down. Those type of variables are given the cunning name of "counter". So if you ever look at other peoples script and see the letter i you will know that it is being used to count something. The other letters that are traditionally used are j and k. Having said all that, well typed in any case, you can use any name for a variable that makes sense to you.

i <= 1152 -- The second argument in the loop is essentially the brakes. This will stop the loop from running the moment that i is no longer less than, or equal too 1152.

This second part of the loop is very important to get right as it is possible to create a loop that never ends. I will give an example of this in a moment.

i++ -- This is a shorthand way writing i = i + 1. Basically all it does is takes the value currently stored in i and adds 1 to it. You can use either method. So every time the loop runs i is incrementing by 1 every time this loop runs.

When I am writing up a loop I am thinking these things exactly: start i at 1, do the loop while i is less than or equal to 1152, and add 1 to i every time the loop... loops

The loop runs any script that is placed between the curly braces.

Loops that work and loops that don't.

It can be annoying when you are using an application like Flash and it stops working either completely or for a period of time. Make a mistake with a loop and this is exactly what will happen. The mistake that you are trying to prevent is a loop that will never end.

Here is an example of a loop that never ends, Don't type this into Flash, all it will do is make the program hang for awhile:

```
for(i=0; i>=0; i++){

    //DO SOME STUFF

}
```

In the example above I have created a loop that starts at 0 will continue looping if i is greater than or equal to 0 and each time the loop runs it will add 1 to i. Therefore i will always be greater than 0 and the loop will never end.

Here is an example of a loop that also will not work:

```
for(i=100; i<=100; i--){

    //DO SOME STUFF

}
```

This time the loop counts backwards, if you look at the code you should be able to see that i is always going to be less than 100. i-- is the opposite of i++.

Here is an example of a loop that will never run, it won't make the program hang or crash it will just do nothing:

```
for(i=0; i>50; i++){

    //DO SOME STUFF

}
```

This is essentially saying start i at 0 and only run the loop if i is greater than 50. As the loop is starting at a value that is less than 50 and therefore not greater than the condition of the loop isn't met and the loop doesn't run.

The above loop is a common mistake. I still regularly mix up the greater than and less than symbols. Lets assume that I want the last loop above to run 50 times, below is an example of a loop that will work, and will run 50 times:

```
for(i=0; i<50; i++){

    //DO SOME STUFF

}
```

It's a small change but now instead of the code never running it will run 50 times.