"For" Loops

“For” loops are used to repeat some operation multiple times, for example to do something with a variable, or to call an engine function several times in a row. It is almost the same as the “While” loop.

A “for” loop's header defines 3 parts things: (1) the counting variable, set to an initial value, (2) the loop condition, and (3) the change in the variable's value. These are separated by a semicolon (;). After the header, comes the body of the loop, which defines what is it that should be done on each loop cycle (note: a loop cycle is usually called an *iteration*). It can consist of a single statement, or of multiple statements in a code block.

The structure of a “for” loop:

```
for (counter; condition; step)
    DoSomething();
```

Or, with a code block:

```
for (counter; condition; step)
{
    DoSomething();
    DoSomethingElse();
    WhyNotDoThisToo();
}
```

Basically, the loop uses the counter variable, checks if the condition is true, executes the loop body if it is, and then changes the counter by the specified step. Then it does it all over again, untill the condition evaluates to false.

There are many different ways you can create “for” loops. Here are some examples:

```
void OnStart()
{
        for(int i = ; i <4; i++)
              AddTimer("T" + i, 1.5 * i, "TimerFunction");
}
```

Let's take this apart. “void OnStart()” is the location it is in, which happens when the level starts up.

The “for(int i = 0; i < 4; i++)” runs the loop body (in this case, the loop body adds a timer that will trigger “TimerFunction”) 4 times before the loop breaks, because integer “i” has to be less than 4, as stated in the loop condition.

NOTE: It's an old programming tradtion to start your counters from 0 - maybe this seems counter-intuitive, but it's often more convenient to do so. Just remember, if you see the initial value of the counter variable set to 0, and the condition states i < 4, a loop like this will execute 4 times, *not* 3. Why? Look at how the value changes: i= 0, 1, 2, 3 , (4 doesn't execute, as the condition is not true for i=4).

If you didn't use the “for” loop, it would look like this:

```
void OnStart()
{
    AddTimer("T0", 1.5, "TimerFunction");
    AddTimer("T1", 3, "TimerFunction");
    AddTimer("T2", 4.5, "TimerFunction");
    AddTimer("T3", 6, "TimerFunction");
}
```

All the code above would do exactly the same thing for each other. You can think of the “for” loop as a shortcut. When you use the “for” loop in your script, don't be afraid to use it. You could always go back here and check out how to do it again. When using the “for” loop, you can change whatever value a parameter has when it's in the “for” loop. Use braces ({ }) when the loop's body exceeds one line. Also another tip is that you can use whatever variable you want for the “for” loop, so long that it doesn't interfere with an existing variable that could do stuff that you don't want to happen. Here is and example for how a “for” loop could vary:

```
void OnStart()
{
    for (int x = ; x <= 4; x += 2)
    {
        AddEntityCollideCallback("Player", "ScriptArea_" + x, true, 1);
    }
}
```

Note that the condition here checks if x is *less then or equal* to 4 (⇐ 4). Also note that the x variable is incremented by 2 on each loop cycle (`x += 2` is just shorthand for`x = x + 2`). This means that the loop will body execute 3 times (remember, the counter started from 0, not 1 - so, i = 0, 2, 4).

You can also loop backwards. For example, assuming you have a long corridor with 10 lights made in the editor, and that the player is closest to corridor_light_0, and that the corridor_light_9 is the farthest, this loop will create an interesting effect, by fading out the corridor_light_9 first, and then working it's way back toward corridor_light_0, increasing the fadeout time (the last parameter) in each step.

```
for (int i = 9; i >= ; i--)
{
    FadeLightTo("corridor_light_" + i, , , , 1, -1, 9 - i);
}
```

I also want to say that in the scripts I provided don't include the functions that could be created based on what I have in there, like a timer function and entity collide functions.

This wiki entry has been made by Kyle S. If you have any comments or need help with this, send me a private message on the Frictional Games Forum. (My name on there is Kyle)

- Edited by Xiphirx. Fixed errors in code and in text.

- Edited by TheGreatCthulhu. Added the introduction, some additional info here and there,

formatted code where required.