Repeat loops:

How an application performs a repetitive task

When writing an application, you'll often want to repeat a task. To do this you use a repeat loop. Revolution offers several different types of repeat loop for different purposes. Let's take a look at a simple loop you would use to create an egg timer.

Video Errata: The video omits the script "put false into gCancelLoop" from the Start Timer button, when adding this global variable. This line should be added after the "end repeat" script line. This resets the gCancelLoop handler allowing the button to be used again.

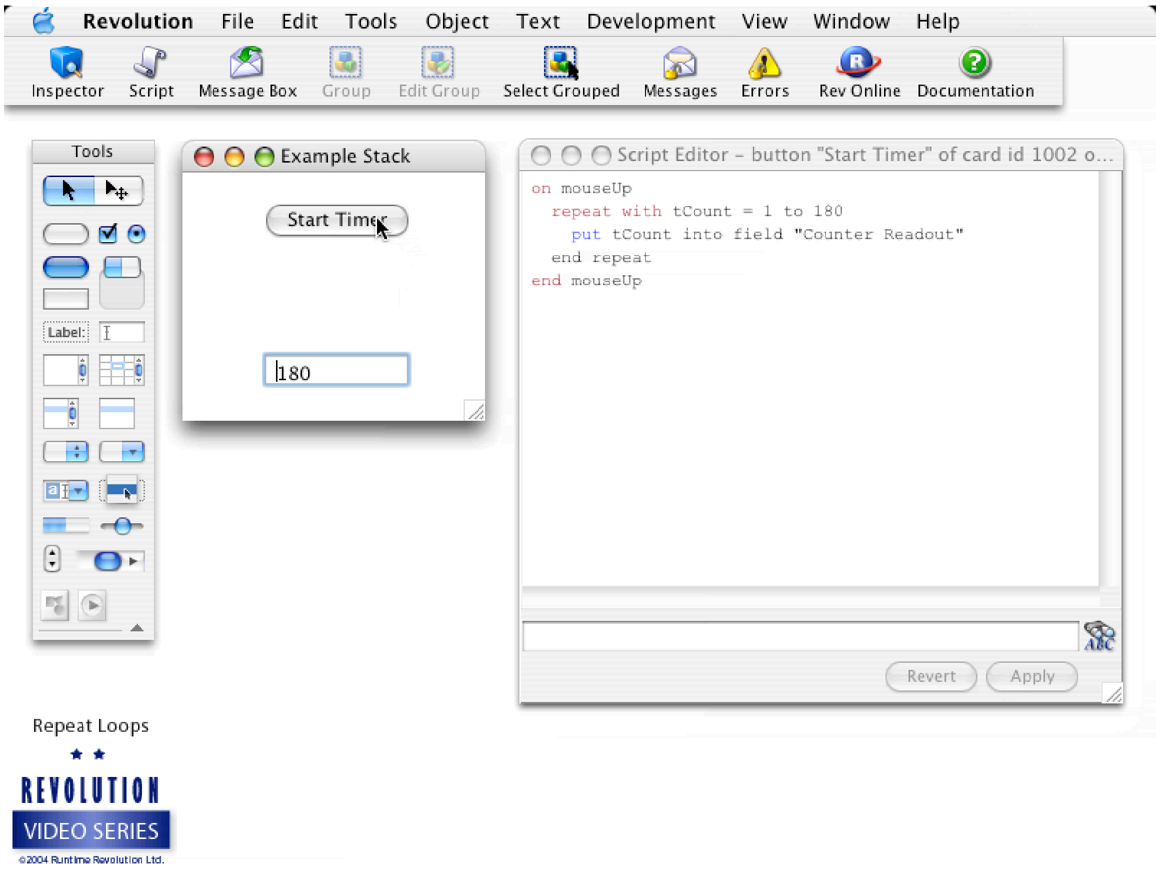
Key topics covered in this tutorial

- One of the key features of a computer is to be able to do repetitive tasks automatically
- Revolution offers a number of different ways to do a repeat loop – to execute the same code several times
- Example of using a repeat loop to count to 180
- How to interrupt a script that is running
- The different types of loop and when to use each one
- Automating tasks in the background using timers

Let's create a button to start the egg timer, and a field to display the time elapsed. We'll name the field 'Counter Readout' and edit the script of the button. We'll start with:

```
repeat with tCount = 1 to 180
  put tCount into field "Counter Readout"
end repeat
```

In this example, we are using a form of the repeat loop called 'repeat with' that increments a variable, in this case tCount, by one, with each repetition of the loop.



```
on mouseUp
  repeat with tCount = 1 to 180
    put tCount into field "Counter Readout"
  end repeat
end mouseUp
```

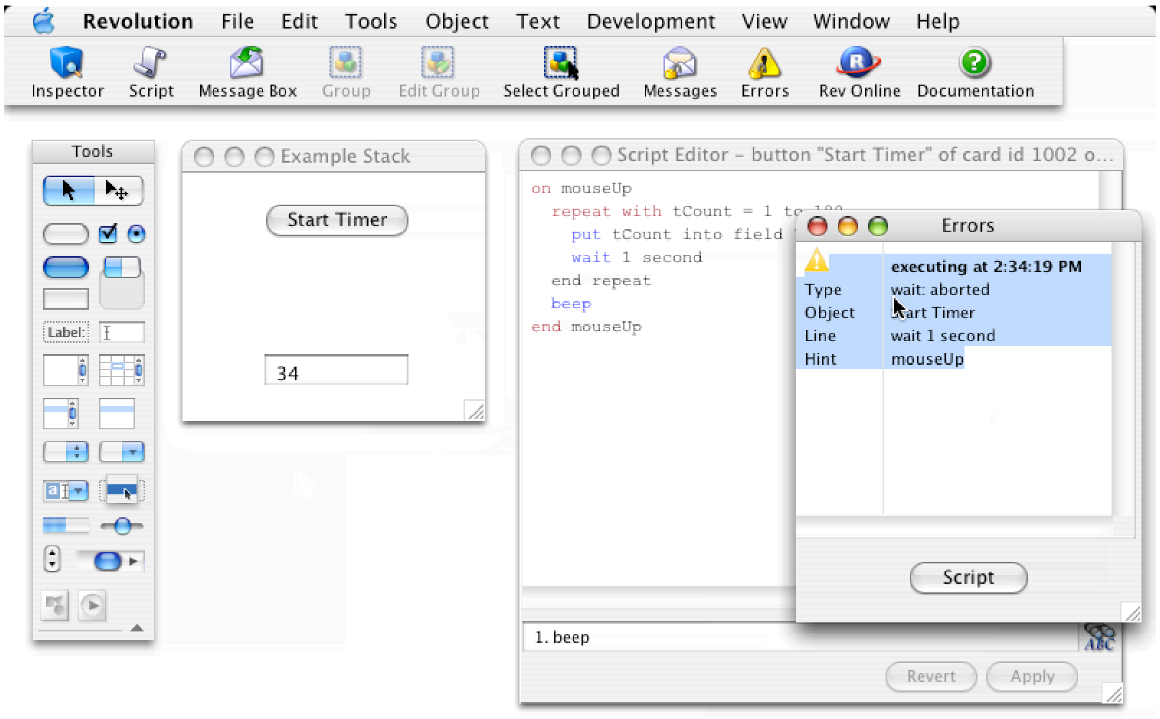
Well, that works, except that it is too fast to see – Revolution is counting as fast as it can, not once every second. We'll use the wait command to slow the script down:

```
wait 1 second
```

By getting the script to wait one second each time round, the whole loop should now take 180 seconds to run, that's three minutes. Let's make this example a little more useful. Let's say that we wanted to time our egg, and have the computer beep at the end of the timer. Let's edit the script and add a beep when the repeat loop is done.

It looks like it's going to work. The only problem we have now is that we're going to have to wait 180 seconds to get to the end of the loop. There is no way to stop the timer – we forgot to add that!

This is a good time to introduce a fallback in Revolution – pressing control-period or, if you are using a Mac, command-period. Pressing this key combination will halt a script that is running, unless you have set a special property that prevents you from interrupting scripts like this.


Revolution File Edit Tools Object Text Development View Window Help
Inspector Script Message Box Group Edit Group Select Grouped Messages Errors Rev Online Documentation
Tools
Example Stack
Start Timer
34
Script Editor – button "Start Timer" of card id 1002 o…
Errors
Script
1. beep
Revert Apply

```
on mouseUp
  repeat with tCount = 1 to 100
    put tCount into field
    wait 1 second
  end repeat
  beep
end mouseUp
```

	executing at 2:34:19 PM
Type	wait: aborted
Object	Start Timer
Line	wait 1 second
Hint	mouseUp

Repeat Loops
REVOLUTION
VIDEO SERIES

The script stops, and we have an error window which says 'wait: aborted', to tell us that we interrupted the script during the wait command. Let's make the script a bit more user-friendly and have it stop when the mouse is pressed. The mouseClick() function will return the value 'true' if the mouse has been clicked since the script started running. So we insert:

```
if the mouseClick is true then exit repeat
```

We can now exit our repeat loop when we click. That's much more user-friendly!

Of course, our script still has one drawback. It takes over Revolution while it is running, blocking access to anything else that might be in your application. Ideally, we would want the script to run in the background. There is a really simple way to change the repeat loop to tell Revolution to allow other scripts to run. Remember that in this loop most of the time is taken in the 'wait' statement. We saw how fast the script ran without that – hardly any time at all. Revolution lets you modify the wait command to allow other things to run. We just need to add 'with messages' to the end of the wait statement to allow messages to continue to be delivered while the script is waiting. Of course, that means that we can't use the mouseClick() function because we want to allow the user to use the mouse. So instead let's change it to wait for a variable to be set before cancelling the repeat loop. We'll add a global variable called gCancelLoop to the script and change

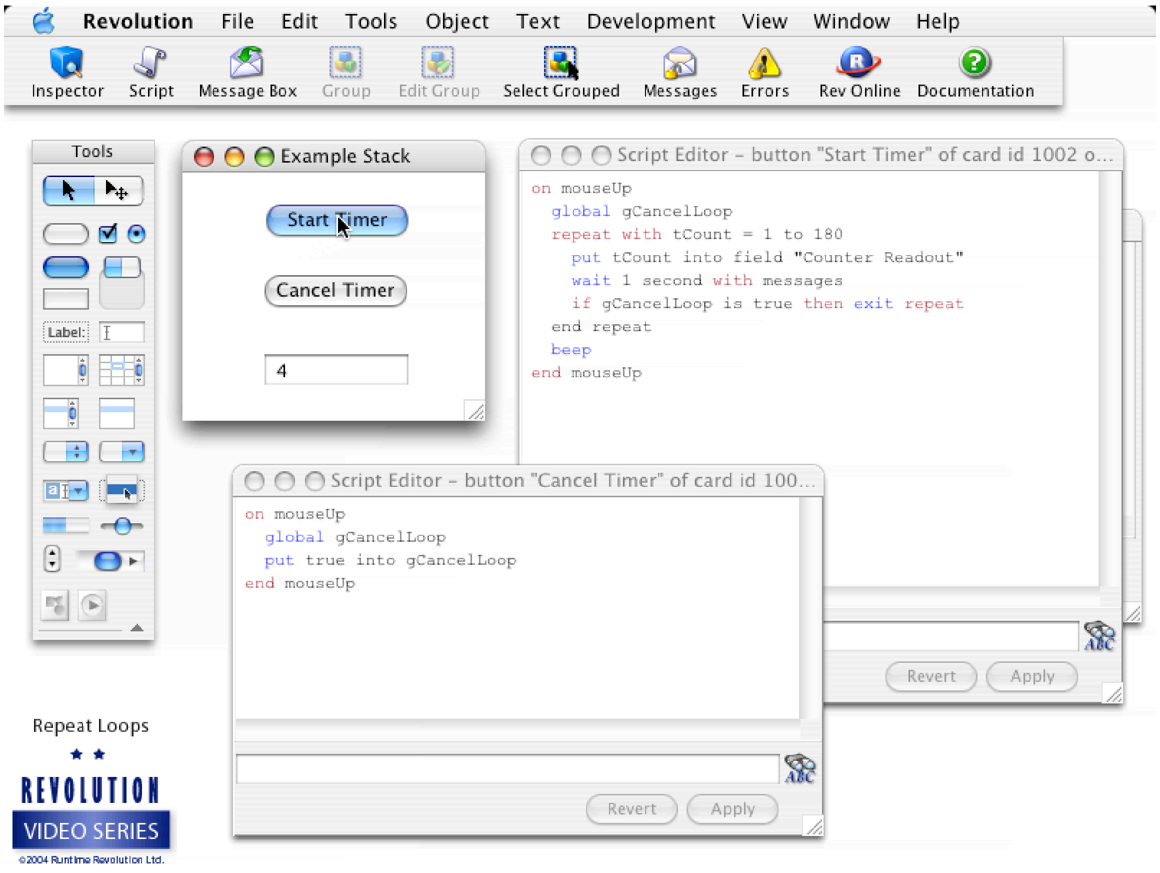
```
if the mouseClick is true then
```

to

```
if gCancelLoop is true then
```

Finally, we need to create another button that will set this variable to true and therefore cancel the repeat loop. We'll call this button 'Cancel Timer' and in the button script we will set the global gCancelLoop to true when the button is clicked.

```
global gCancelLoop
put true into gCancelLoop
```



Script Editor – button "Start Timer" of card id 1002 o...

```
on mouseUp
  global gCancelLoop
  repeat with tCount = 1 to 180
    put tCount into field "Counter Readout"
    wait 1 second with messages
    if gCancelLoop is true then exit repeat
  end repeat
  beep
end mouseUp
```

Script Editor – button "Cancel Timer" of card id 100...

```
on mouseUp
  global gCancelLoop
  put true into gCancelLoop
end mouseUp
```

We can now interact with other objects and if we want to cancel the timer all we have to do is press the ‘Cancel Timer’ button and the repeat loop exits.

If our task was more complex or if we had multiple tasks that we wanted to do in the background, or if our task was all based on processing and didn’t contain any 'wait' statement, we would need to use a different method to allow the task to run in the background. We cover that method in detail in the tutorial on timers.

Appendix: Scripts used in this tutorial

```
on mouseUp
  repeat with tCount = 1 to 180
    put tCount into field "Counter Readout"
  end repeat
end mouseUp
```

```
on mouseUp
  repeat with tCount = 1 to 180
    put tCount into field "Counter Readout"
    wait 1 second
  end repeat
  beep
end mouseUp
```

```
on mouseUp
  repeat with tCount = 1 to 180
    put tCount into field "Counter Readout"
    wait 1 second
    if the mouseClick is true then exit repeat
  end repeat
end mouseUp
```

```
on mouseUp
  global gCancelLoop
  repeat with i = 1 to 180
    put i into field "Counter Readout"
    wait 1 second with messages
    if gCancelLoop is true then exit repeat
  end repeat
put false into gCancelLoop
end mouseUp

on mouseUp
  global gCancelLoop
  put true into gCancelLoop
end mouseUp
```