Lua Scripting Timer und EventSequence - LUA

Inhaltsverzeichnis

Timer und EventSequence

Seit Version 1.0.2 ist nun auch das Einsetzen eines Timers und einer EventSequence möglich. Mit EventSequence können wir Ereignisse in zeitlicher Reihenfolge auslösen.

1 Timer

Um ein Ereignis mit einem Timer auszulösen, brauchen wir zunächst einen Verweis auf die Timer.lua Modul Datei, die sich seit Version 1.0.2 in dem Unterordner "Engine" befindet.

```
local Timer = require("Engine.Timer")
```

Anmerkung:

Diesen setzen wir am Besten ganz oben in unsere Level Lua Datei, damit er bei jedem Speichern und Laden

```
---- FILE: \TEN_102.lua
local EventSequence = require("Engine.EventSequence")
local Timer = require("Engine.Timer")

LevelFuncs.OnLoad = function() end
LevelFuncs.OnSave = function() end
```

Dann benötigen wir noch einen Auslöser, den wir in eine LevelFuncs. Funktion setzen. Dann den Aufbau für den Timer.

Code

```
local Timer = require("Engine.Timer")

LevelFuncs.KillLara = function()
Lara:Explode()
PlaySound(106)
PlaySound(30)
end

LevelFuncs.StartTimer = function ()
local myTimer = Timer.Create("my_timer", 5.0, false, {minutes, seconds, milliseconds}, LevelFuncs.KillLara)
myTimer:Start()
end
```

Alles anzeigen

Hier haben wir in die "**LevelFuncs.KillLara**" folgende Ereignisse gesetzt:

Lara explodiert.

Es wird der Explosionssound 106 abgespielt.

Es wird der Lara Schrei Sound 30 abgespielt.

Den Timer haben wir in eine "**LevelFuncs.StartTimer**" Funktion gesetzt.

Wir weisen dem Timer eine lokale Variable mit dem Namen "myTimer" zu.

Wir erstellen einen Timer mit dem Namen "my_timer" und sagen nach 5 Sekunden soll die "LevelFuncs.KillLara" ausgeführt werden.

Code

```
{minutes = false, seconds = true, deciseconds = true}
```

Hier können wir einstellen, ob Minuten, Sekunden oder Dezimalsekunden angezeigt werden sollen. "False" heißt immer "nein" und "True" heißt immer "ja".

Durch "myTimer:Start()" wird der Timer ausgelöst, sobald wir die "LevelFuncs.StartTimer" in einem VolumeTrigger auslösen.

Die Funktion "LevelFuncs.StartTimer" können wir jetzt in einen Volume Trigger setzen.

Unter Timer Module finden wir noch mehrere Funktionen die wir setzen können wie. z.B. den Timer zu pausieren, zu stoppen usw.

2 EventSequence

Um Ereignisse mit zeitlicher Reihenfolge mit einer EventSequence auszulösen, brauchen wir zunächst einen Verweis auf die EventSequence.lua Modul Datei, die sich seit Version 1.0.2 in dem Unterordner "Engine" befindet.

```
local EventSequence = require("Engine.EventSequence")
```

Anmerkung:

Diesen setzen wir am Besten ganz oben in unsere Level Lua Datei, damit er bei jedem Speichern und Laden

```
1 ---- FILE: \TEN_102.lua
2 local EventSequence = require("Engine.EventSequence")
3 local Timer = require("Engine.Timer")
4
5 LevelFuncs.OnLoad = function() end
6 LevelFuncs.OnSave = function() end
```

Dann benötigen wir noch die Funktionen in einer "LevelFuncs. Funktion", die wir zeitlich hintereinander auslösen wollen. Dann den Aufbau für die EvenSequence.

Code

```
local          EventSequence          =          require("Engine.EventSequence")

LevelFuncs.opendoor1                =                function                ()
local              gate              =              GetMoveableByName("door_1")
gate:Enable()
end

LevelFuncs.opendoor2                =                function                ()
local              gate              =              GetMoveableByName("door_2")
gate:Enable()
end

LevelFuncs.opendoor3                =                function                ()
local              gate              =              GetMoveableByName("door_3")
gate:Enable()
end

LevelFuncs.TriggerSequence              =              function              ()
local             mySeq             =             EventSequence.Create("my_seq",
false,
--{seconds = true, deciseconds = true}, -- timer format, see Timer for details
10,                                                                  LevelFuncs.opendoor1,
5,                                                                   LevelFuncs.opendoor2,
3,                                                                    LevelFuncs.opendoor3
)

mySeq:Start()
end
```

Alles anzeigen

Als Ereignisse haben wir hier im Beispiel drei LevelFuncs. Funktionen erstellt, die hintereinander 3 Türen öffnen.

Die Funktionen haben die Funktionsnamen "**LevelFuncs.opendoor1**", "**LevelFuncs.opendoor2**" und " **LevelFuncs.opendoor3**".

Die EvenSequence haben wir in eine "**LevelFuncs.TriggerSequence**" Funktion gesetzt.

Wir weisen der Eventsequence eine lokale Variable mit dem Namen "mySeq" zu.

Wir erstellen eine EventSequence mit dem Namen "my_seq". Durch "false" legen wir fest, dass nach Ende der EventSequence diese sich nicht wiederholt. Bei "true" gibt es eine Endlosausführung.

```
{seconds = true, deciseconds = true}
```

Hier legen wir das Format fest, in welchem die Zeitangabe erfolgen soll. "Sekunden, Dezimalsekunden usw." Ähnlich wie beim Timer oben.

Anmerkung:

Möchten wir zwischen den einzelnen Ereignissen keinen Timer anzeigen, setzen wir einfach alle Werte auf "false", z.B.

```
{seconds = false, deciseconds = false}
```

Nach 10 Sekunden, lösen wir nun die Funkion "LevelFuncs.opendoor1" aus.

Danach nach 5 Sekunden die Funktion "LevelFuncs.opendoor2".

Danach nach 3 Sekunden die Funktion "LevelFuncs.opendoor3".

Mit mySeq:Start() wir die EventSequence ausgelöst, wenn wir die "LevelFuncs.TriggerSequence" in einem Volume Trigger auslösen.

Unter EventSequence Module finden wir noch mehrere Funktionen die wir setzen können wie. z.B. die EventSequence zu pausieren, zu stoppen usw.

Wichtige Anmerkung:

In der EventSequence.lua Datei in Version 1.0.2 wurde ausversehen nicht auf die aktuelle Timer.lua

```
38 --       mySeq:Start()
39 --   end
40 --
41 -- @luautil EventSequence
42
43 local Timer = require("Timer")
44
45 local EventSequence
46
47 LevelFuncs.Engine.EventSequence = {}
48 LevelVars.Engine.EventSequence = {sequences = {}}
49
```

muss man durch:

```
local Timer = require("Engine.Timer")
```

ersetzen.

Alternativ habe ich hier auch eine korrigierte EventSequence.lua angehängt.

EventSequence.lua