Verwendung von Alarmen

Description

Alarme haben wir bereits in mehreren Tutorials verwendet. Es wird nun Zeit, dass wir uns mit dem Thema näher befassen.

Was sind Alarm?

Erst einmal müssen wir uns in Erinnerung rufen, dass der GameMaker Event basiert arbeitet. Das heißt, dass bestimmte Ereignisse ausgelöst werden, sobald ihre Bedingung erfüllt wurde ohne sie einfach an der Reihe sind. Eine Bedingung kann sein, dass ein Objekt erstellt wird. Dafür gibt es das **Create-Event**. Zu Bedingungen, die an der Reihe sind, gehören u. A. Step und Draw. Die laufen immer, in einer vom GameMaker festgelegten Reihenfolge.

Alarme sind Events, die man aufrufen muss und zwar inklusive einer Zeitangabe. Das ist der wesentlichste Unterschied zu Skripten und gibt uns damit auch besondere Möglichkeiten. So kann man zum Beispiel eine Instanz im Spiel generieren, sie aber nach einer gewissen Zeit automatisch zerstören. Alarme können dabei nicht nur vom eigenen Objekt sondern auch von anderen Objekten aufgerufen werden. Das heißt, jedes Objekt hat Zugriff auf die Alarme anderer Objekte, wie auch jedes Objekt auf Variablen anderer Objekte zugreifen kann.

Wie verwende ich Alarme?

Wie bereits erwähnt, werden Alarme per Zeitangabe aufgerufen. Das heißt nicht, dass wir dem Alarm sagen, er solle um 17:30 loslegen, sondern nach wie vielen Steps nach dem Aufruf er starten soll. Wie schnell die Steps laufen,wird beim Speed im Raum eingestellt. Wenn da 60 steht, dann ergibt eine Zeitangabe von 60 genau eine Sekunde. Wenn im Raum 60 steht und wir den Alarm mit 10 Steps aufrufen, dann startet er nach 0,167 Sekunden.

Die zweite Sache die wir wissen müssen ist, dass es sich beim Aufruf eines Alarms genau genommen um ein Array handelt. Das heißt, dass die Nummer des Alarms in einer eckigen Klammer stehen muss. Genau genommen kann jedes Objekt bis zu 12 Alarme verwenden, die von 0 bis 11 durchnummeriert

sind. Wenn wir einen Alarm als Event erstellen möchten, geben wir die entsprechende Nummer an und müssen auch diese aufrufen.

Der konkrete Aufruf für Alarm 0 sieht so aus:

```
alarm[0]=60;
```

Das heißt, dass alles, was in Alarm 0 ist, in 60 Steps gestartet wird. Wenn wir mit 60 eine Sekunde meinen, weil wir die Raumgeschwindigkeit auf 60 gestellt haben, können wir das auch so schreiben:

alarm[0]=1 * room_speed;

Im Alarm selbst kann ich den eigenen Alarm ebenfalls aufrufen, oder andere Alarme. So kann man auch eine Schleife erzeugen, was ich bereits im Schleifen-Tutorial gezeigt habe.

Wichtig ist, dass ein Alarm nicht mehrfach zur gleichen Zeit aufgerufen wird, weil man ihn damit immer wieder neu startet. Wer also aus dem **Step-Event** einen Alarm aufrufen möchte, muss unbedingt sicherstellen, dass sich nach dem Aufruf die Bedingung ändert. Dazu ein kleines Beispiel: Angenommen, ein Alarm soll ausgelöst werden, wenn eine bestimmte Bedingung erfüllt wurde. Diese Bedingung wird im Step-Event geprüft und der Alarm von hier aus gestartet.

```
if (Bedingung)
{
     alarm[0] = 10;
}
```

Dann wird der Alarm nie gestartet, so lange die Bedingung erfüllt ist, weil der Alarm mit jedem Step neu aufgerufen wird. Man muss also entweder die Bedingung zurück setzen oder, falls die Bedingung noch für andere Dinge gebraucht wird, eine zweite Variable verwenden.

```
if (Bedingung1) && (Bedingung2)
{
     alarm[0] = 10;
     Bedingung2 = false;
}
```

So wird der Alarm nicht weiter gestartet und die erste Bedingung ist weiterhin gegeben.

Wie kann ich ohne Alarme arbeiten?

Alarme sind eine sehr praktische Sache und man löst damit auch gerne Dinge, die dafür gar nicht so gut geeignet sind. Manchmal werden Alarme als Code-Container genutzt, die dann vom eigenen oder von anderen Objekte aufgerufen werden. Man sieht das vor allem dann, wenn ein Alarm mit einem Step aufgerufen wird und der Alarm weder sich selbst, noch andere Alarme aufruft.

Manche verwenden Alarme auch als Zeitmesser. Das ist durchaus legitim, aber etwas ungenau. Alarme reagieren auf die Raumgeschwindigkeit und diese kann bei Spielen auch einmal niedriger liegen als die vorgegebene Speed. Das ist der Fall, wenn der Computer des Spielers nicht schnell genug ist, oder man schlicht die Fenster wechselt und wieder zurück kommt. In vielen Fällen mag das

keine Rolle spielen, doch wenn man ein genaues Timing braucht, sind Befehle wie *current_time* besser. Das werden wir in einem eigenen Tutorial behandeln.

Wer generell auf Alarme verzichten will, muss mit zusätzlichen Variablen und dem **Step-Event** arbeiten. Die Theorie dazu sieht so aus: Man braucht zwei Variablen. Die erste Variable zählt die Steps, ist also ein Counter, die zweite Variable stoppt den Counter.

Event-Create

counter = 60;

stopCounter = false;

Wir haben den Counter erstellt und die zweite Variable definiert. Nun zum Step-Event:

```
if (!stopCounter) && (counter <= 0)
{
     // Hier kommt der Code, den man sonst im Alarm hätte
}

if (counter > 0)
{
     counter--;
} else {
     stopCounter = true;
}
```

Spätestens hier merkt man, wie praktisch Alarme sind. Wenn man weitere Dinge wie Steuerung, weitere Abfragen, Positionsänderungen etc. im Step verwendet, kann das ganz schön unübersichtlich werden. Deswegen ist es in solchen Fällen auch sinnvoll, möglichst viel Code in Skripte auszulagern.

Date Created
11. November 2016
Author
sven