20.3.0 Timer

Die Klasse *Timer (gb)* implementiert ein Timer-Objekt. Ein Timer-Objekt ist als Taktgeber aufzufassen, der mit jedem Takt Ereignisse auslöst. Die Zeit zwischen den einzelnen Takten wird vom Wert der Eigenschaft *Delay* festgelegt (Taktzeit in Millisekunden).

20.3.0.1 Eigenschaften

Die Klasse *Timer* besitzt nur zwei Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Delay	Integer	Gibt die Anzahl von Millisekunden zwischen den Timer-Ereignissen zurück oder setzt die Anzahl von Millisekunden zwischen den Takten.
Enabled	Boolean	Startet den Timer mit *Timer.Enabled* = *True* oder ermittelt, ob der Timer aktiv ist. Ein de-aktivierter Timer kann keine Timer-Ereignisse mehr auslösen.

Tabelle 20.3.0.1.1 : Eigenschaften der Klasse Timer

20.3.0.2 Methoden

Die Klasse *Timer* hat drei Methoden, deren Beschreibung Sie in der folgenden Tabelle finden:

Methode	Beschreibung
Start	Der Timer wird gestartet. Das hat den gleichen Effekt wie *Timer.Enabled* = *True*.
Stop	Der Timer wird gestoppt. Das hat den gleichen Effekt wie *Timer.Enabled* = *False*.
Trigger()	Timer.Trigger() löst das Timer_Timer()-Event bei der nächsten Iteration der Event-Schleife aus. Diese Methode ist nützlich, um den Event-Handler "später" auszuführen – mindestens nach dem aktuellen Stack von Funktionsaufrufen oder zum nächsten expliziten Aufruf von WAIT.

Tabelle 20.3.0.2.1 : Methoden der Klasse Timer

20.3.0.3 Ereignis

Für die Klasse *Timer* existiert genau ein Ereignis → *Timer_Timer()*. Dieses Ereignis wird jedes Mal ausgelöst, wenn die über *Timer.Delay* festgelegte Taktzeit abgelaufen ist.

Beachten Sie, dass die Taktzeit *mindestens* Timer.Delay beträgt. Es besteht im Allgemeinen – abhängig von der verwendeten Hardware und und dem Betriebssystem und seiner Konfiguration – keine Garantie für eine „zeitlich nahe" Ausführung des Event-Handlers.

20.3.0.4 Einsatzfelder für die Klasse Timer

Es folgt eine Zusammenstellung von ausgewählten Einsatzfeldern für Timer-Objekte aus unterschiedlichen Demonstrationsprojekten im Online-Buch:

- Uhren (analog, digital, binär),
- Stopp-Uhren,
- Kurzzeitmesser,
- Count-Down-Zähler (Datum oder Zeit),
- TimeOut bei zeit-kritischen Routinen,
- Auslösen von Aktionen einmalig, mehrmalig (asynchron) oder periodisch → Messwerte, Status-Erfassung, automatische Speicherungen,
- Repeat-Funktionen für Button,
- Blinkende Anzeigen von Text oder von Symbolen (einfarbig oder mehrfarbig) mit einstellbarer Frequenz.

Dieses Konzept hat beim Einsatz eines Timers bewährt:

- Feststellen, ob der Einsatz eines Timers das vorgegebene Ziel erreichen lässt.
- Takt-Zeit ermitteln und die Eigenschaft *Timer.Delay* mit einem geeigneten Startwert festlegen.

- Festlegen in welchen Prozeduren der Timer gestartet und gestoppt wird.
- Festlegen, ob das 1. Timer-Event *sofort* nach dem Timer-Start mit der Methode *Timer.Trigger()* ausgelöst werden soll oder erst *nach* dem 1. Takt.
- Genau definieren, was passieren soll, wenn das Ereignis *Timer_Timer()* ausgelöst wird.

20.3.0.5 Timer-Funktion Timer()

Die Timer-Funktion *Timer()* – die in keinem Zusammenhang mit der Klasse Timer steht – gibt die Anzahl der Sekunden an, die seit dem *Programm-Start* vergangen sind. Der Funktionswert ist vom Datentyp *Float*. Die Funktion benutzt die System-Uhr.

Mit diesen Quelltext ...

```
Dim fZeitdifferenz as Float

fZeitdifferenz = Timer
Print "Sekunden seit Programm-Start = ";;fZeitdifferenz;;" Sekunden"
```

… wird in der Konsole der Gambas-IDE die Anzahl der Sekunden angezeigt, die seit dem Programmstart vergangen sind:

```
Zeit seit Programm-Start =  17,8296689987183  Sekunden
```

Die Funktion kann zum Beispiel zur Laufzeitmessung beim Vergleich von Such- oder Sortieralgorithmen oder bei ausgewählten, zeitintensiven Prozeduren eingesetzt werden.