Fortgeschritte Inhalte für AntIT!

1.) Erweiterung von Bedinungen

Das Grundgerüst einer if-Abfrage kann durch einige Funktionen erweitert werden und damit für viele Anwendungen nützlich sein.

Als erstes können nachgeschaltet else- und if-else-Blöcke angehängt werden. Nähere Informationen finden sich hier:
https://www.w3schools.com/js/js_if_else.asp

Außerdem kann der Ausdruck für die Bedingungen selber mit weiteren Vergleichen (insbesonderen != für Ungleichheit) und logischen Verknüpfungen (insb. && (und), || (oder) und ! (nicht)) kombiniert werden. Näheres dazu hier:
https://www.w3schools.com/js/js_comparisons.asp

Die Links enthalten ausführliche Beispiele und ein paar kleine Übungen.

2.) Variablen

Innerhalb eines Events können für temporäre Werte Variablen verwendet werden. Diese werden mit var initialisiert und können dann mit ihrem Namen aufgerufen werden. Einführende Informationen finden sich hier:
https://www.w3schools.com/js/js_variables.asp

Als konkretes Beispiel, wie man das in AntIT! verwenden kann:

```
17 Ameise.wenn("Wartet", function(){
18     var entfernung = Distanz(Position, Bau)
19     entfernung = entfernung + 200
20     if (entfernung < 1000) {
21         GeheZuBau()
22     }
23 })
```

In Zeile 18 wird eine Variable ‚entfernung' definiert und mit einem Wert belegt. In Zeile 19 wird dieser Wert um 200 erhöht. In Zeile 20 verwenden wir diesen Wert innerhalb inner if-Abfrage.

Variablen sind nur innerhalb des Events gültig!

3.) Gedächtnis

Um Informationen über verschiedene Runden hinweg für eine einzele Ameise zu speichern, gibt es für jede Ameise ein Gedächtnis-Objekt. Dieses Objekt mit über die Punktschreibweise mit Attributen erweitert werden, die über verschiedene Runden erhalten bleiben. Ein einführendes Beispiel:

```
1  var Ameise = AntIT.NeueAmeise("Cmeise")
2
3  Ameise.wenn("IstGeboren", function(){
4      Gedächtnis.Name = Zufallsname()
5      Gedächtnis.Zähler = 0
6  })
7
8  Ameise.wenn("Wartet", function(){
9      Gedächtnis.Zähler++
10     console.log(Gedächtnis.Name + " wartet " + Gedächtnis.Zähler + " mal")
11     Gehe(200)
12     Drehe(20)
13 })
```

Bei der Geburt werden zwei Einträge erstellt: Einmal ein Name und ein Zähler. Attribute müssen nicht deklariert werden, sondern werden bei der ersten Zuweisung automatisch erstellt. Daher ist es wichtig, alle verwendeten Gedächtnis-Einträge bei Geburt mit einem Wert zu belegen.

Die Funktion „Zufallsname()“ erstellt bei jedem Aufruf einen zufälligen Ameisennamen.

In Zeile 9 wird der Zähler-Eintrag um 1 erhöht. In Zeile 10 werden dann die Informationen zur Konsole ausgegeben. Starte eine Simulation mit dieser Ameise und drücke **F12**. Dann erscheinen die entsprechenden Nachrichten.

Die Namen der Einträge sind frei wählbar, sie verhalten sich sonst so wie Variablen.

4.) Sichtung mit ID

Die Events SiehtZucker und SiehtApfel können mit einem zweiten Parameter erweitert werden, der zum gesichteten Objekt noch eine eindeutige ID bereitstellt. Damit können Nahrungsmittel von verschiedenen Ameisen als gleich erkannt werden. Beispiel:

```
8  Ameise.wenn("SiehtZucker", function(zucker, id){
9      console.log(id) // z.B. 4
10 })
```

5.) Nachrichten senden

Wichtige Information zum Verständnis: Die Ameisen haben ein gemeinsames, telepathisches Band und können über das gesamte Spielfeld miteinander kommunizieren. Das unterscheidet sie grundlegend von „echten“ Ameisen.

Mithilfe der Funktion „SendeNachricht()“ können Ameisen über das Spielfeld mit den anderen Ameisen kommunizieren. Dazu nimmt die Funktion als Parameter den Namen der Nachricht. Die anderen Ameisen erhalten daraufhin ein Event, das aus dem Namen besteht und einen vorangestellten Doppelpunkt:

```
8  Ameise.wenn("SiehtZucker", function(zucker){
9      SendeNachricht("zuckerGesehen")
10 })
11
12 Ameise.wenn(":zuckerGesehen", function(){
13     // andere Ameise empfängt Nachricht
14 })
```

Die erste Ameise sieht einen Zucker und sendet über Zeile 9 eine Nachricht mit dem Titel „zuckerGesehen“. Andere Ameisen erhalten die Nachricht, wenn sie auf das Event „:zuckerGesehen“ warten. Achtung: Selbst wenn beide Events in einer Ameise stehen wird der Code von *verschiedenen* Ameisen ausgeführt!

Nachrichten können mit einem Parameter erweitert werden:

```
8  Ameise.wenn("SiehtZucker", function(zucker){
9      SendeNachricht("zuckerGesehen", zucker)
10 })
11
12 Ameise.wenn(":zuckerGesehen", function(zuc){
13     GeheZuZiel(zuc)
14 })
```

Dieser Parameter wird mit der Nachricht an die anderen Ameisen übertragen.

Die Nachrichtenevents (hier als „:zuckerGesehen“) werden von nah nach fern sortiert aufgerufen. Nahe Ameisen erhalten die Nachricht also früher.

6.) Selber Nachrichten senden

Zur Strukturierung des Codes kann es nützlich sein, sich selber Nachrichten zu schicken. Das funktioniert mit dem Befehl „SendeSelber()“. Beispiel:

```
3  Ameise.wenn("Wartet", function(){
4      GeheZuBau()
5      SendeSelber("bauErreicht")
6  })
7
8  Ameise.wenn("bauErreicht", function(){
9      console.log("hurra")
10 })
```

Nachdem der Bau erreicht wurde, wird das Event in Zeile 8 aufgerufen. Nachrichtenevents an sich selber werden nicht mit einem Doppelpunkt geschrieben.

Sich selber Nachrichten zu schicken ist vor allem im Zusammenhang mit dem Gedächtnis wichtig. Zuweisung in das Gedächtnis werden immer sofort zum Zeitpunkt des Events ausgeführt:

```
3  Ameise.wenn("Wartet", function(){
4      Gedächtnis.Status = "Rückweg"
5      GeheZuBau()
6      Gedächtnis.Status = "Angekommen"
7  })
```

Dieses Programm ist fehlerhaft. Während des Rückwegs soll der Status auf „Rückweg“ sein. Erst wenn die Ameise im Bau angekommen ist, soll der Status auf „Angekommen“ gesetzt werden. In diesem Fall wird der Status im Moment von Wartet ausgeführt, d.h. der Status lautet „Angekommen“, obwohl die Ameise sich noch gar nicht zum Bau bewegt hat. Die richtige Lösung lautet:

```
3  Ameise.wenn("Wartet", function(){
4      Gedächtnis.Status = "Rückweg"
5      GeheZuBau()
6      SendeSelber("bauErreicht")
7  })
8
9  Ameise.wenn("bauErreicht", function(){
10     Gedächtnis.Status = "Angekommen"
11 })
```

7.) Sterben

Ameisen können durch Wanzen sterben oder wenn sie ihre Reichweite von 3000 Schritten aufgebraucht haben. Dazu gibt es ein eigenes Event:

```
5 Ameise.wenn("IstGestorben", function(ursache){
6     alert(ursache)
7 })
```

Die Ursache ist entsprechend entweder „Wanze“ oder „Müdigkeit“.

Die Reichweite kann im Bau aufgeladen werden.

8.) Rand

Ebenso gibt es ein Event für den Fall, dass die Ameise an den Rand stößt:

```
3 Ameise.wenn("RandErreicht", function(){
4     Drehe(180)
5 })
```

9.) Zufall

Es können Zufallszahlen erzeugt werden und damit die Bewegung randomisiert werden:

```
9 Ameise.wenn("Wartet", function(){
10     Drehe(Zufallszahl(-30,30))
11 })
```

Die Ameise dreht sich dann um -30 bis 30 Grad.

Die Funktion liefert Ganzzahlen. Sie kann eigentlich für jeden Zweck verwendet werden.

10.) Aktivität

Wenn man Zucker und Äpfel gespeichert hat, kann man über die Funktion „Aktiv()“ überprüfen, ob sich das Objekt noch auf dem Spielfeld befindet oder schon abgebaut ist:

```
9 Ameise.wenn(":zucker", function(zuc){
10     if (Aktiv(zuc)) {
11         GeheZuZiel(zuc)
12     }
13 })
```

AntIT! für Ameisenspezialisten

"Most good programmers do programming not because they expect to get paid or get adulation by the public, but because it is fun to program."

- Linus Torvalds

10.) Globale Variablen

Wenn Variablen außerhalb eines Events definiert werden, dann gelten diese für alle Ameisen und den gesamten Ameisenbau. Man stelle sich vor, was für ein mächtiges Werkzeug das ist: Zustände können über den gesamten Ameisenbau gespeichert werden und jede Entscheidung kann mit Hilfe des Wissens um die gesamte Situation getroffen werden. Das ist so, als ob der Ameisenbau plötzlich einen eigenen „Geist“ hätte und anfangen kann zu sehen. Globale Variablen sind also die Elemente, mit denen sich die einzelnen Informationen der Ameisen sammeln lassen und sich zu einem Gesamtbild fügen können.

Ein erstes Beispiel. Wir wollen die Anzahl der Ameisen zählen. Dazu definieren wir eine Zählervariable. Diese ist eine normale, nur dass sie außerhalb des Events geschrieben wird. Dann kann ganz normal auf diese Variable in allen Events von allen Ameisen jederzeit zugegriffen werden:

```
1  var Ameise = AntIT.NeueAmeise("Dmeise")
2
3  var bevölkerung = 0
4
5  Ameise.wenn("IstGeboren", function(){
6      bevölkerung++
7      console.log("Es leben " + bevölkerung + " Ameisen.")
8  })
9
10 Ameise.wenn("IstGestorben", function(){
11     bevölkerung--
12 })
```

Natürlich ist der nächste verlockende Schritt, eine Datenstruktur zu erstellen, die Buch führt über gesichtete Nahrungsmittel und dazu die Positionen speichert! Dann können die Ameisen ihre Arbeitskraft optimal entfalten. Ein wesentlicher Schritt ist noch die Einführung von Objekten:

https://www.w3schools.com/js/js_objects.asp

To be continued ...